Requirement Specification

Version 3.0

November 20, 2020

Accepted as baseline requirements for this project:

Client:	Chris Doughty	11/20/2020
	Signature	*Date*
Team Biosphere:	Wesley Smythe	11/20/2020
	Signature	*Date*
		11.20.2020
	Signature	*Date*

Project Sponsor: Chris Doughty
Team Mentor: Andrew Abraham
Team Members: McKenna Chun, Gregory Geary, Wesley Smythe, Chufeng Zhou, and Kainoa Boyce

Table of Contents

1 Introduction **2**

2 Problem Statement **4**

3 Solution Vision **5**

4 Project Requirements **6**

4.1 Functional Requirements 6

4.2 Performance Requirements 11

4.3 Environmental Requirements 13

5 Potential Risk **15**

6 Project Plan **17**

7 Conclusion **19**

Figure 0: Roaming Tiger

1 Introduction

Tropical forests are biodiverse hotspots filled with many species of flora, fauna, and fungi. Many of these species in recent years have become extinct, or are at risk of extinction due to an influx of human activity in the wild. The root of this current mass extinction lies in issues that have been unresolved for years, these include:

- Climate Change
- Illegal Deforestation / Logging
- Illegal Poaching

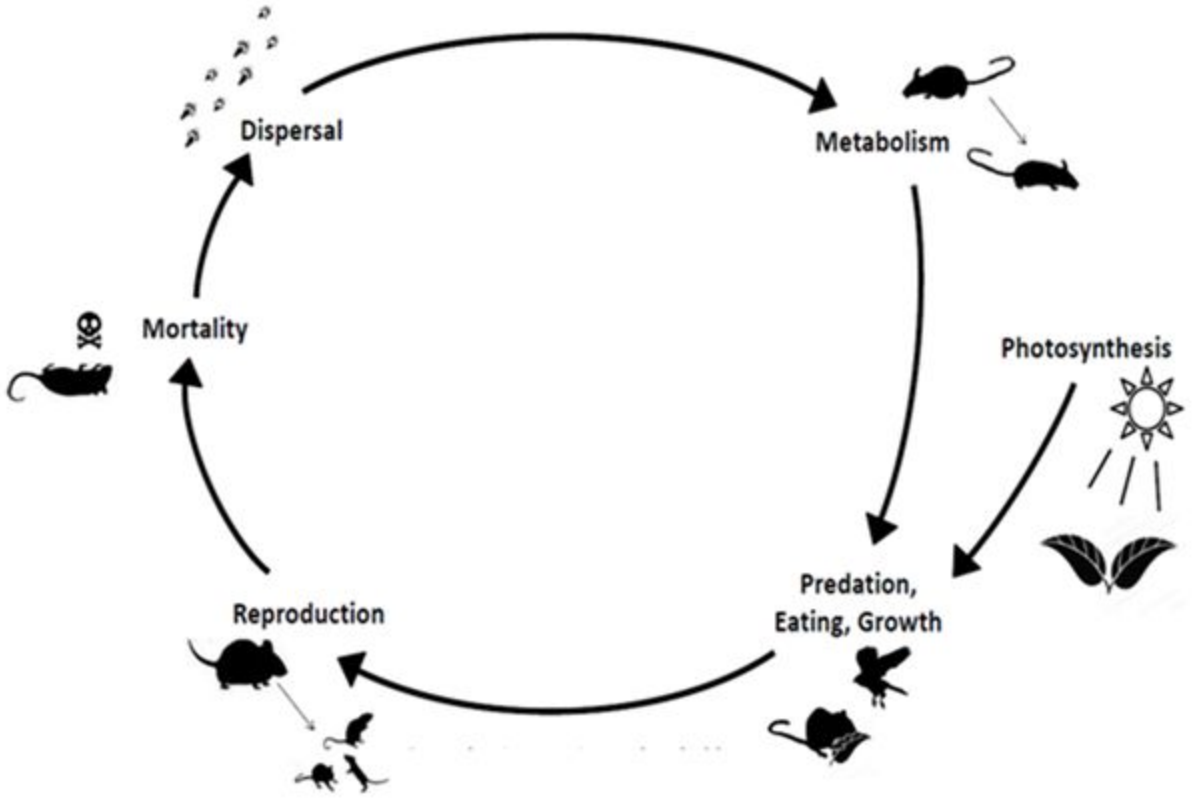
While these are not the only problems, they are some of the most well known and afflict most of the damage to biodiversity in tropical forest regions. Many scientists around the world have dedicated themselves to researching and collecting information on these topics. From this, many solutions and models have been created and attempt to remedy the predicament that tropical forests and other vulnerable areas are in.

Between all of the different suggestions made, a recent model has become known in the biodiversity research community: the Madingley model. Models like this are used to generate a general view of human impact on the environment, but the Madingley model is different. It possesses revolutionary capabilities that allow it to surpass other existing models, such as being able to represent all life on Earth.

It also includes the interactions between different species and the environment, which can be specified for each ecosystem. With this unique feature, it allows for changes over time in the interactions between species, since it is not a fixed concept as defined in other similar models.

The data that the user inputs are also very specific factors like the amount of flora and fauna in an ecosystem. This inputted data can also be altered to test many of the different scenarios from current issues. For instance, the aforementioned logging or poaching could be represented by a decreased input of flora or fauna respectively. Using the interactions in the Madingley model (Fig. 1) below, it is clear that a significant loss to any of the elements would cause a disruption to the entire ecosystem cycle.

Figure 1.0: The Madingley Model Interactions
Source: Madingley Model Github Repository[1]



The output data shows how these situations affect the region, whether they do not change much, or lead to the extinction of even more species, the model is able to predict such an outcome and display simplistic but clear data. While the output of this model is easy to understand, the calculations are much more complex and require professionals in the field to guarantee accurate data.

One such professional is the sponsor of this project: Dr. Chris Doughty, an assistant professor and leader of the Megabiota research laboratory at Northern Arizona University. Dr. Doughty has focused his research on the ecological topic of megafauna, which include large animals and their roles in ecosystems.

In his research, he has worked with the Madingley model to run several simulations based on gathered data. This data is tested with different scenarios, and the output allows scientists like Dr. Doughty to identify the outcomes of the test using the raw datasets or graphing programs like NASA's Panopoly. This analyzed data is then used for research papers and projects by Dr. Doughty and the Megabiota laboratory.

[1] https://madingley.github.io

2 Problem Statement

2.1 Current Workflow

The NAU Megabiota laboratory is able to run the Madingley model with different scenarios and data input to see the effects that may occur. The datasets that are outputted allow researchers like Dr. Doughty to examine and draw results from them.

This data is then incorporated into other projects by Dr. Doughty and the rest of his laboratory, such as scientific research papers or projects. In these papers, the numeric data is “translated” from the raw data produced by the model, and turned into words or visualizations for readers.

2.2 Current Problems

Issues that are present in the Megabiota laboratory research process can be focused around the two core topics:

- Distribution of the data
- Visualization of the data

For the distribution of this data, it is currently used for aiding in research purposes and in scientific papers. This does not allow for a wide range of audiences aside from other individuals in the same scientific field. The limitations on the distribution of this data can be further narrowed down into the following topics:

- Website is a small public platform
- Website can only be accessed from web clients
- Research papers typically read by only other researchers
- Graphical data can only be obtained from programs like Panopoly

As for the visualization of data, the Madingley model produces immense amounts of data for places all around the world. Although Dr. Doughty is focused primarily on the tropical forest regions, just the tropical forests hold a lot of data. The problems with visualization can also be narrowed down to the following topics:

- Tables of raw data are unappealing
- Current graphics have no interaction with users

Through a combination of these two problems, the data being produced by the Madingley model is not being properly introduced to the public as intended. Therefore these issues need to be remedied in order for the data from the revolutionary model to be fully utilized.

3 Solution Vision

The general solution to this problem is to create an application that allows the target audience to visualize the results of the Madingley model. This describes the basic idea of how to solve the problem. In order to guide the end-user through the application, a handful of features must be identified.

- The user will be able to select a location based off of their own location or based off of a location selected on a map.
- The user will be able to select a predefined Madingley Model scenario. The exact scenarios for this application have not been identified by the client. However, some examples include: bushmeat hunting, climate change, and deforestation.
- The user will be able to choose a visualization style to display tailored results from the rendered scenario.
- The user will be able to export the rendered data.

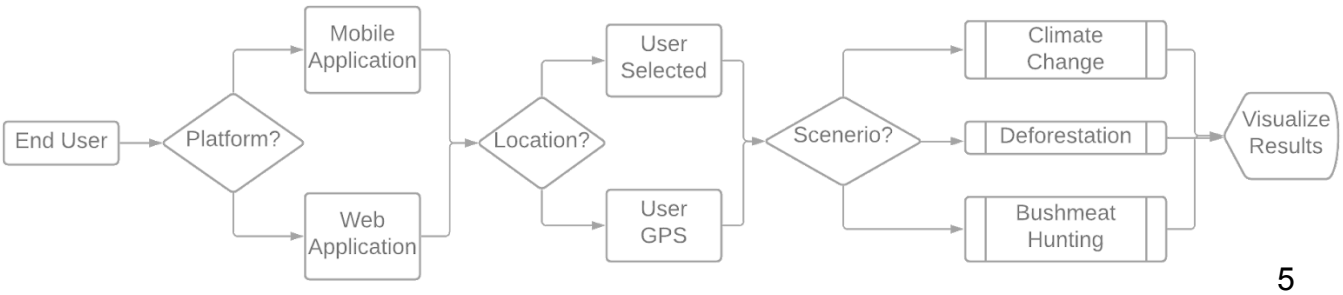
In order to allow the user to select some location, the application will need access to the user's GPS, or access to an interactive map object. In both situations the user will have selected some location of interest. This location selection will allow the application to hone in on the geographical area of interest.

The user will then be asked to choose a scenario from a predefined list of scenarios, this list could include: climate change, deforestation, and bushmeat hunting. The scenario will allow the application to narrow down what dataset should be considered.

Given the selection of the above mentioned choices, the application will then be able to return a small finite set of data that will be of use to the end user. The final steps remaining in the solution plan involve the user selecting what type of output they wish to view. This option will then allow the application to tailor the scenario results to a pre-specified sub audience. This specification is done to allow the user to quickly and easily view the information of interest rather than a general output.

Once the user has viewed the rendered data, the final step that the user will be interested in, involves exporting the information. This step will allow the user to save the results of a scenario to be used later as they see fit, whether it be for further analysis or for presentation purposes.

Figure 3.0: General outline of our solution.



4 Project Requirements

In the following section, we will be going over all of the known requirements for our application. These requirements will be broken down into three categories: functional requirements, performance requirements, and environmental constraints. The first of which are the functional requirements which are the operations that our application will be able to perform such as being able to utilize the user's GPS location, or the ability to send and retrieve data relevant to the Madingley model to be displayed by the applications UI. These will be followed by the performance requirements which are the characteristics that aren't functionally necessary, but are needed to ensure the application is up to industry standards, and can execute its functions in an efficient and safe manner. Some of these would include the speed of the application and its security. Lastly, we will cover the environmental constraints of our application. These are the constraints that have been put on by our team, mentor, and client for various reasons such as the platform compatibility, or previously built components that are to be built into the application. All of which have been initially put together through various meetings with our team, mentor and client, and have been further refined through intensive research and discussions.

4.1 Functional Requirements

The application will require numerous features to operate, this section will outline the baseline functions that will be required for the application to be operational. The following specifications have been outlined in a hierarchical fashion spanning from most important to least important.

- Data Storage
    - The ability to store relative application and user data amongst a designated cloud system
- Graphic User Interface (GUI)
    - The application will be viewed in a coherent and easy to understand way.
    - The main menu of the application allow the user to select from a handful of options to learn more about the project, technologies used, and project sponsor(s)
    - The application will provide for various biodiversity options that the user can check off, and will be displayed by the simulation
- Location access
    - Access to the user's GPS.
- Data processing
    - A cloud based data processing back end that can interact with both the stored data and the client side application.

- Visualization
    - The ability to visualize the results of a given scenario in the form of heatmaps, graphs, and informational tables
    - The ability to tailor the visualization to predefined subgroups like “policy makers” and “general public”.
- Data Exportation
    - The application will have the ability to export the results of a scenario in a raw (CSV) form as well as collection of images representing tables and graphs generated from the scenario.

4.1.1 Graphic User Interface (GUI)

The first functional requirement of our application is a graphic user interface. This will be the primary component connecting the user to the valuable data provided by the madingley model. The GUI will be developed using the Ionic framework since that was the framework of choice for developing our Progressive Web Application. This GUI will be broken down into the following sub requirements:

- Intuitive Overall Design - New user’s should be able to easily navigate throughout the application, and have no issues accessing any of its primary features.
- Login Page - There must be a login page in order to distinguish policy makers vs. general users
- Main Menu - There must be a main menu that allows the user to navigate between key features and information about our team and project. Some of the menu items listed include:
    a. Run Simulation
    b. Simulation Info
    c. Project Info
    d. Team Info
- Interactive map - Upon choosing to run a simulation, the user will then be prompted to choose a location. This will be done a few different ways, the first of which would be for him/her to manually enter their desired location. They will also have the option to use their device's current GPS location, or select it from a map. This is where the interactive map comes into play. User’s should be able to navigate throughout the map and place a circle or square on their chosen location representing the total area for which the simulation should be performed on.
- Input Options - After the user selects a location they will then be able to check off different options which will determine the form of output they will be looking at (deforestation, future predictions, bushmeat hunting etc).

Figure 4.1.1 below is a screen chart that models the general flow of the GUI we plan to create.

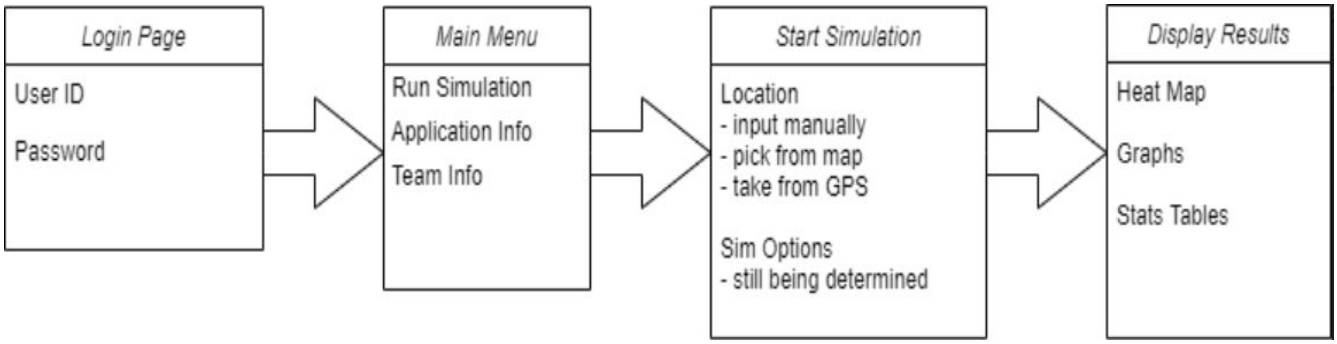


figure 4.1.1 - Application UI Flow Chart

4.1.2 Hardware Access: GPS Services

The fundamental purpose of our app is to allow a user to input a location and various options, and then display a visualization of the statistics output that is accurate according to the Madingley model. Since a location input is required, it is usually standard that the user has the ability to manually input a location, or use the device's GPS location. Therefore it is necessary that our application has the ability to utilize the user's GPS location of the user via whatever device they are using (Web Browser, Android, iOS). This will be done using Ionic's built in GPS services package called Geolocation.

4.1.3 Data Processing

Data processing will be another key functional requirement since various amounts of data must be sent to and from the GUI to be stored in the cloud. This will start after the user inputs their location and various simulation options into the GUI on his/her device, it will be necessary to send that data to a cloud server due to fact which will then be inputted into the spin up of the model itself. Since the application will be utilized from a mobile device or web browser, all of the heavy lifting will need to be done by serverless functions in the cloud.

The output data will need to be filtered or shrank down since the data sets returned by this model are typically between 5 and 10 GB and not at all intended to be displayed on a mobile device or web browser. This will be done by serverless cloud functions that after their done formatting and filtering the data, will then send the remaining necessary data back to the GUI to be shown to the user by the various UI display components.

The figure 4.1.3 below shows how this will be done exactly:

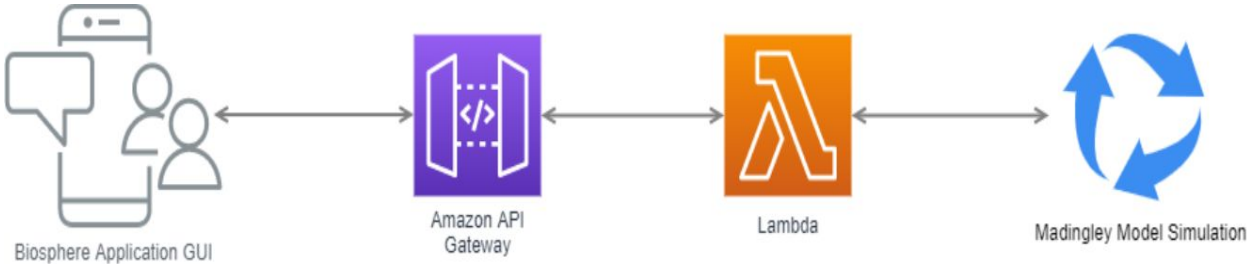


figure 4.1.3 - Application Data Flow Chart

As shown above, data will come from the user's input into the UI and/or will be pulled using the previously mentioned Geolocation package. It will then be sent to one or more AWS Lambda Functions via the API gateway, where it will be inputted into a spin up of the Madingley Model Simulation. From there, the data output of the simulation will then be passed back to the application UI where it will be displayed using various charts and such.

4.1.4 Visualization

The visualization of data will primarily be used by in-built javascript graphic libraries that will allow for user interaction. Users will be able to change this visualization by selecting specific areas on a map based on either a selected location or based upon the GPS location of the user. The specific visualization of data will also be based upon the data that is provided to the application, as less data will cause a smaller visualization of data, and vice versa.

4.1.5 Data Exportation

The exportation of data in the application will primarily be done in a few ways using built in libraries to bounce data between our AWS databases and the Ionic application. One such way is by using the mailto function in Ionic's Javascript library, this can be primarily used to request data from the AWS database and sent to the user's application. This provides a quick and easy methodology to transfer data without complication, as it does not have to worry about data security, due to the data that we are transferring in this section is the data that wants to be shown to people.

While the mailto function can serve as the primary source of transferring data, it does not have many security measures against data leaks. Therefore for much more secure data transfers like user authentication, the AWS Simple Email Service, abbreviated as

SES will be used. AWS SES allows for a secure transfer due to its many built in security features including:

- End-to-End Data Encryption
    - Allows for data to be secure even if intercepted by using protocols built by amazon such as S/MIME or PGP which cause the data to be useless if intercepted
- Virtual Private Cloud Usage (VPC)
    - Shared computing resources created by Amazon for the specific reason of transferring data between their services
- Receiver Authentication using Sender Policy Framework (SPF)
    - Helps to avoid data leaks by preventing the forgery of a sender address, as it flags transfers that are attempting to use the expected address, but do not belong to it
- Sender Authentication using Domainkeys Identified Mail (DKIM)
    - Checks that the place that data is being transferred to matches the correct domain key that was provided before, to avoid middle-man interceptions of data

Through the combination of these two features of our services, we are able to create both efficient and secure data transfers when necessary. As they are both interchangeable with one another, the transfer of non-private data may be used by one or both of the specified methods.

4.2 Performance Requirements

This section outlines characteristics of the application as well as the entire system. These characteristics are used to standardize the experience for all users as well as ensure the application remains operational and functional with as few limitations and hindrances as possible.

4.2.1 Global Access

Although the immediate target audience is located in the Americas, Gabon, Indonesia, and across the tropics, it is important not to limit the possibility of all users to these regions. The end goal of this application is to be used around the world. Given that goal, the initial creation of this application should reflect that goal. As such components should be hosted in redundant configurations and have servers located around the global. Since the developers associated with this project are located solely in the United States, this requirement cannot be tested, but it can be assumed based on documentation provided by AWS, the App Store, and Play Store.

4.2.2 Responsiveness

This application provides the target audience with important information that cannot be found elsewhere, as such, this application must be able to handle large amounts of traffic and maintain a base level of responsiveness. There should be little to no screen lag between application pages that do not rely on back-end components. If a new page does rely on a back-end component there should not be a significant (10+ second) lag for processing. In order to prevent this, code should remain simple and non-repetitive. If it is determined that the sequential variation of the code cannot be improved upon then the code base will be parallelized to the best of its ability. These steps should ensure a responsive application that will be able to deliver the necessary information to the end user in order to prevent delay in their research or policy related works.

4.2.3 Security

A typical development cycle focuses on implementing features as fast as possible without consideration for the security of the features. The development cycle of Team Biosphere will include a risk assessment prior to the addition of any new features. Although the user will not be storing or inputting any information that is considered sensitive, it is important to consider the security of application to prevent malicious actors from hijacking sessions, or feeding malicious packets into the end-user’s connection. In order to ensure the security of the end-user, steps will be taken to

authenticate and limit the traffic and protocols accepted by both the server and client sides.

4.2.4 Application Crashes

Applications that crash or fail frequently have a high likelihood of losing users to its lack of usability. In order to limit changes of crashes there the application will run based on lazy processing. This means that the application will only process content on a need by need basis. This can limit the resources and runtime associated with the application. This can be during the development process by keeping track of how many times the application crashes at each iteration. If the number of crashes increases as the iterations increases then the application is flawed at a very low level.

4.2.5 Language Options

A stretch goal associated with this project involves the ability to support multiple languages. This constraint, or feature allows the application to reach non english speakers. This requirement can be tested by copy-pasting the application text into some translator(s) and determining if the output reflects the intended message.

4.3 Environmental Requirements

In order to tailor the solution to the specified needs of the client beyond the basic functionality and performance requirements, a handful of constraints have been imposed by the client, team, as well as the software used.

4.3.1 Programming Language Constraints

As a result of the selected SDK, and other system components, we have been limited to what programming languages we will be able to use to develop our application. This limitation causes our application to be susceptible to the benefits and pitfalls of the specific languages chosen. The two languages that are anticipated to be used are Javascript and Python.

Javascript, while simple and easily interpreted by the underlying system, has a number of unintended consequences that must be considered. The first being its lack of browser support. Every browser has its own interpretation of javascript and as such, applications are subject to possible broken or missing features. This is especially true for outdated or less popular browsers like Internet Explorer, Opera, GNOME Web. The overall effect of this constraint is unknown since we do not know what browser or their associated versions, the target audience will be using. Assuming that they will be using up to date and modern browsers, then this concern could be close to non existent.

The primary back-end language will be Python. Python will be primarily used to handle the fetching, simplification and some visualizations of the application. Although Python, similar to javascript, is a popular language it has key downfalls that may limit the responsiveness of the application. This stems from the fact that Python is an interpreted language rather than a compiled or machine level translated language. The limitations of an interpreted language can be lessened by using basic parallelization techniques like multiprocessing or multithreading. That being said, the limitations of Python should also be limited in the MVP stages. The limitations of interpreted languages could have an effect on the responsiveness of some of the stretch goals, namely the ability for the end user to make use of a toy variation of the Madingley. These limitations should be considered given that the Madingley model is computationally expensive, regardless of how simple the model may become.

4.3.2 Web and Mobile Application

In order to reach the largest number of people with our application, the client has requested that this application run on at least two mobile platforms as well as a web platform. In order to accomplish this without the need for duplication or restructuring of code, it was determined that the best solution to this constraint would be to develop a

progressive web application (PWA). This will for the development of both an iOS and android mobile application as well as a web browser variant that can be used by the browser. However, this constraint limits the scope of the solution to web based components and mobile based components. Due to the hardware limitations and dependencies of both mobile and web based components, the amount of computation done on an end user's device must be limited to the capabilities and components that are built-in to the device.

5 Potential Risk

Given that this project relies on system components that are dissimilar from each other there is room for risks, as such, these risks should be managed. A failure that occurs at a crucial location could spell disaster for some or all of the connected elements. In order to mitigate and properly assess the risks an each component and its dependencies should be considered and single points of failure (SPoF) should be avoided whenever possible.

5.1 System calculation error

Likelihood：Extremely-low
Severity： High
There does not exist any application that has absolute precision, this application is no exception. This application relies heavily on calculations done within the Madingley Model as well as its interpretation for visualization. This calculation error could occur because the Madingley Model is written in C#, the application back-end is in python and visualized to the end user in Javascript. As a result of their varying levels of floating point precision the results of a given scenario could be prematurely rounded. The overall precision can range from 29 digits to 5 digits depending on the datatypes used at each stage. Premature rounding on this scale means that any digit less than 0.00001 cannot be completely trusted.

5.2 AWS Service Disruption

Likelihood：Extremely-low
Severity： High
Given that all back-end files and functions will be stored in some form of AWS cloud service, it is important to make note of the likelihood of these resources being inaccessible. According to AWS's internal status checker, there has only been a single large scale outage in AWS's history. That being said, about 50 users a day report some sort of down AWS resource. This is extremely tiny in comparison to the total number of users and resources. If the AWS services associated with this application experience a disruption, this would cause the application to stop working until service is restored.

5.3 Security

Likelihood：Unknown
Severity：　High
There are numerous attack vectors since this application relies on several moving parts. Given the large attack surface there is a high likelihood for some level of attack, however given that all resources are managed by trusted external parties the likelihood is low. However, if the attack surface has a common vulnerability or severe exploit then botnets and scrapers around the world could accidentally stumble upon this application and cause major security issues. Given the above factors it is not possible to determine the exact level of risk associated with this application. Once the attack surface is analyzed and proper steps are taken to prevent intrusions, injection, or hijacking then the likelihood of a security threat that could compromise the target audience, stakeholder, or developers can be minimized.

6 Project Plan

Our team has accomplished a lot since we originally started this project. As seen in figure 6.0.1., one of the main things we have accomplished so far is refining our MVP. In addition to that task, we also created a rough model of the application, developed a better understanding of the Madingley Model, and researched possible services that we could use for our solution.

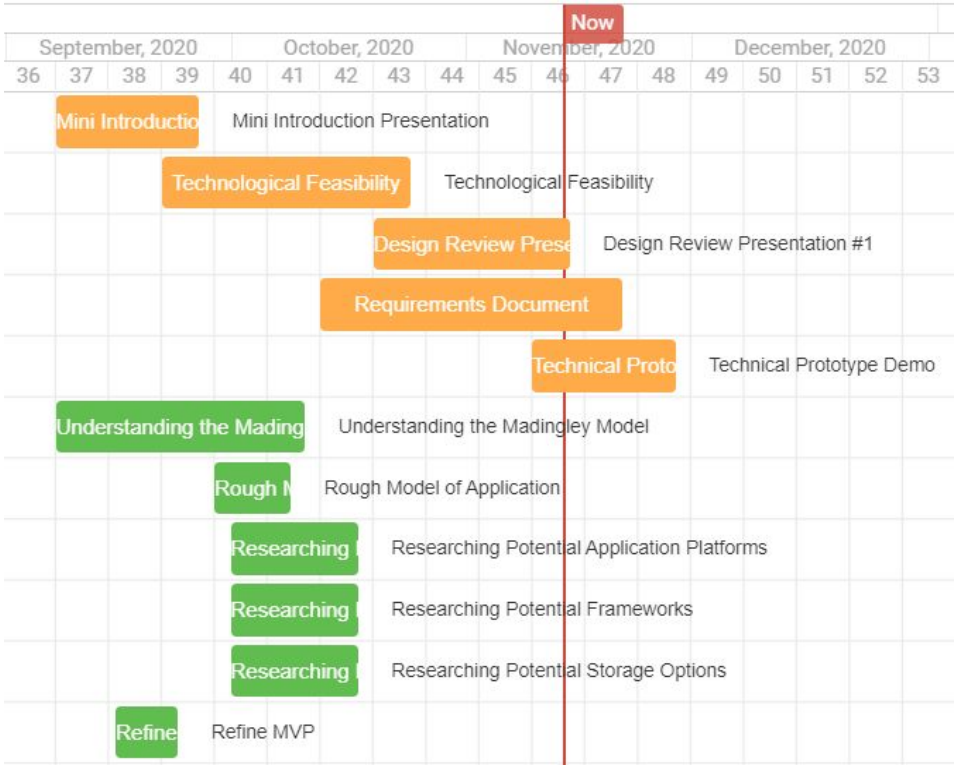


Figure 6.0.1: Gantt chart representing the Fall 2020 semester. The orange tasks represent course-based assignments. The green tasks represent our team driven tasks.

As the first part of our project comes to a close, we are optimistically looking toward the second half of our project. This stage will start during our winter break. Over winter break, we are planning on gaining more experience with Ionic and AWS. At the conclusion of break, we will be ready to complete the next steps of our project. The first step our team is looking forward to is the development of the front end interface. The main requirement that we will be adding first will be the implementation of a basic Graphic User Interface. By doing this, we will have a working application for Android, iOS, and the web.

Then, we will implement the code needed for data processing and location access. This will require the use of databases, as seen in figure 6.0.2. Our team will also implement the Madingley Model simulations, so that the user has a better way to visualize the data. We won't have to create the simulations. Instead, we will create a

better way to visualize the data from the simulations. In this version, we will also implement the ability for the user to select an area on which they want to run the simulations. This was one of the main requirements emphasized by the client, so it will be vital for us to implement this at this part of the development process. Next, we will add the back end of the program. The back end will give us the ability to process information from both the user and the stored data.

Finally, we will create a full prototype that will have all the features we decided on for the MVP. If time permits, we will also try to include a couple of the stretch goals into this prototype. The first stretch goal that we plan on implementing is a multiple language feature. The user would be able to use our product from a small group of languages. If we get that implemented, the next feature we hope to add is providing the user with the ability to export the output. This would allow the user to save information and eliminate the need for them to rerun the program every time they want to visualize the same data. However, before we can implement those features, we will have to make sure that all the features of the MVP work almost perfectly. There might be a few areas of the code that need to be perfected, but 90% of the project will be completed once we have finished the full prototype. Lastly, we will fine tune and put the finishing adjustments on our prototype, so that we can deliver the final product to our client.

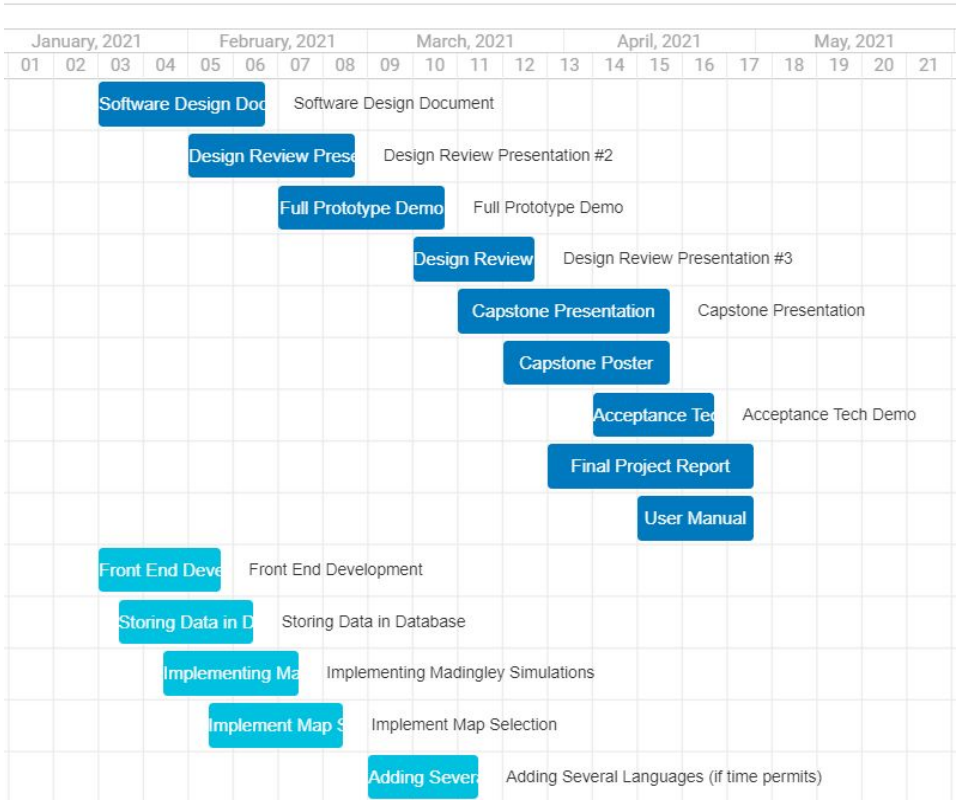


Figure 6.0.2: Gantt chart representing the Spring 2021 semester. The dark blue tasks represent course-based assignments. The light blue tasks represent our team driven tasks.

7 Conclusion

The goal of this project is to provide information derived from Madingley Model scenarios. The Madingley model is an ecological model that has the ability to model biodiversity for the entire planet. It was determined that our target audience includes the general public, specifically those interested in the decline and status of biodiversity on the planet; as well as policy makers. Policy makers are interested in this information in order to make more environmentally conscious decisions for the planet.
In order to accomplish this goal with as little hindereraces as possible, a progressive web application (PWA) will be created in order to reach individuals on both iOS, android, and personal computer browsers. This will prevent our target audience from being inherently hindered due to a lack of platform support.
This document outlines some of the key functionalities and considerations taken to ensure a secure and positive experience for the end user. Some of these functionalities include:

- A security and response-time based development strategy.
- Word wide access to the application, assuming the target audience has internet access.
- The ability to choose a location of interest either by use of the user's GPS or by pinpointing a location on a map.
- The ability to tailor the output to their predefined preferences.
- The ability to export the data in both a visualized and raw form.

While this document does serve as a pseudo-formal contract between both the client and development team as to what expectations and design choices are both realistic and necessary for the success of this project. Its secondary purpose is to indicate that our team is prepared to tackle the oncoming challenges with a plan that the developers and project stakeholders are satisfied with. The next step is to take this plan and turn it into a reliability.