TEAM BIOSPHERE

Software Design Document

Version 2.0

February 12, 2021

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

Table of Contents

Figure 0: Tropical bird[1]

[1]http://auchterphotography.com/tropical-birds-beautiful/

1.0 Introduction

Tropical forests are biodiverse hotspots filled with many species of flora, fauna, and fungi. Thousands of these species in recent years have become extinct, in addition to this, globally there are one million species at risk of extinction[1] due to an influx of human activity in the wild. While the extinction of species is a natural process, the fact that species extinctions are up by one thousand percent since humans have gotten involved[2], is not. The root of this current mass extinction lies in a variety issues that have been unresolved for years, these include:

- Increasing Climate Change from human sources such as man-made pollution
- Illegal Deforestation / Logging
- Illegal Poaching

While these are not the only problems, they are some of the most well known and afflict much of the damage to biodiversity in tropical forest regions. Many scientists around the world have dedicated themselves to specifically researching and collecting information on this topic. From this, many solutions and models have been created in attempt to remedy the predicament that vulnerable biodiverse areas like tropical forests 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 these 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. Another distinguishing feature is its ability to include the interactions between different species and the environment. It 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 into the model are specific factors like the amount of flora and fauna in an ecosystem, which can be altered to test many of the different scenarios from current issues. An example of this could be the aforementioned logging or poaching scenarios being represented by a decreased input of flora or fauna respectively.

The Madingley Model itself is used by researchers such as our client: Dr. Chris Doughty, an assistant professor and leader of the Megabiota research laboratory at Northern

[1] https://www.nationalgeographic.com/environment/2019/05/ipbes-un-biodiversity-report-warns-one-million-species-at-risk/
[2] https://www.nationalgeographic.com/news/2014/5/140529-conservation-science-animals-species-endangered-extinction/

Arizona University. Dr. Doughty and his team at the Megabiota lab have focused their research on the ecological topic of how factors like climate change or poaching can affect an ecosystem, specifically regions consisting of tropical forests.

While researchers like Dr. Doughty and the Megabiota lab are able to produce many results from running the Madingley model on data they have continued to collect. This data includes calculated predictions on the biodiversity of tropical forests in upcoming years based on the different scenarios listed previously. However, the format of the data itself poses a few issues for spreading the information:

- They are in raw data form, and without any assistance from outside resources or programs to visualize them, they are simple datasets with overwhelming amounts of information that are unappealing to view
- Visualization programs such as NASA's Panoply are able to provide a visual representation of the inputted data. However, programs like these are not easy to access, requiring specific platforms to run on and some technical expertise to operate

With these problems in mind, Dr. Doughty has commissioned an application to solve both of these listed issues. In addition to these resolved issues, the application will be able to serve different types of users. For example, those with different roles in society, while a scientist would have different statistical data than a construction worker, they would both be able to use the primary functions of the application. This includes other demographics as well, as the application will function on both iOS and Android, it will be compatible with almost any mobile device, in addition to web clients.

The main functionality will be to allow users to select different ecological scenarios in different regions of the world, and then provide visualizations of the data. Specifically, the application will have the following features and functions:

- The application will run the same on all devices, whether mobile devices that use iOS or Android, or web clients like browsers, it's functionality will be the same
- The user interface will be simple, to allow users easy accessibility to the different features provided
- The users will be able to select different roles, whether they are a scientist or the general public, which will impact the variety of data provided
- The application will allow users to select different locations around the world using a marker placed on the map provided, it will also support the use of the device's GPS location feature with permission to select their current location

- The application will also allow users to select from a variety of predefined scenarios in which will provide the user with previously calculated data using the Madingley model on the region they have selected
- The output data provided to the user will be visualized in a way that permits general understanding and appeals to the user aesthetically

With these features in mind, the final product will provide the user with new knowledge of ecosystems and how specific factors can change them over periods of time. By the time the user is done using the application, the overall goal is to have both educated the user themselves and help to spread awareness of these issues.

2.0 Implementation Overview

The Madingley model is a very complex framework that requires some level of expertise from both the software side, and ecology/biology to understand and operate. However, the findings generated from this model are incredibly valuable. As such, an application must be developed to provide an easy-to-use experience for users who do not have the: money, technical expertise, or computer hardware to run this model.

This project aims to do just that, raise awareness of the importance of biodiversity by presenting semi customized findings from the Madingley model to all users that have an internet capable browser or device.

In order to reach the largest number of users as possible, this project will be generated using a Progressive Web Application (PWA). This is an application that will be viewable by users who have access to an android device, iOS device, or any modern web browser. Given that the solution requires that both the front-end and back-end must be cross platform; the most optimal solution would be to house as much as possible in the cloud. Given that our team has experience with Amazon Web Services (AWS) and some of the available services, it was determined that using AWS as the back-end would be the best course of action. Within AWS the following services will be used:

- **Lambda**: A serverless function used to process application requests, and to parse the Madingley Model results.
- **S3**: A simple storage service used to house all of the scenarios, and their associated data.
- **API Gateway**: An intermediary service that connects the application front-end to the amazon back-end services.
- **IAM**: A policy-based control system used to manage and limit access to resources associated with the production and development environments of the application cycle.

While there exists numerous PWA software developer kits (SDKs) it was determined that Ionic / Ionic-Angular will be used. Ionic was chosen as a result of its general popularity, use of existing languages, and compatibility with AWS. Ionic allows developers to create three applications using a single code base. Ionic is compatible with iOS devices, android devices, and modern web browsers. This limits the amount of time it takes to develop a given application and allows the developers to create a look and feel that is similar across platforms. For more details regarding alternative front-end and back-end components as well as the criteria for each component, see Requirements specification version 3 or the Technical Feasibility Analysis Report.

The last and arguably most important aspect of this project is the visualization associated with the requested data. If this application did not contain any graphical / visualization elements, the user would be left with a table of hundreds of values which would have little to no effect on the end-user. In order to visualize the data requested, it was determined to use native Javascript and angular libraries since they are natively compatible with Ionic, and in turn the various platforms supported by ionic.

3.0 Architectural Overview

In this section, we will discuss the application components and their interactions. Our project will have four main components: AWS S3 bucket, AWS lambda functions, AWS API Gateway, and the Ionic interface. The S3 bucket will store all of the Madingley Model data. In order to retrieve the data, we will use the lambda functions. They will fetch the data and process it. After it has been processed, the API Gateway will transport the data from the lambda function to the Ionic interface. When the data reaches the Ionic interface, the user will get the output that resulted from their initial input. Finally, we will discuss some of the reasons why we selected each of these components.

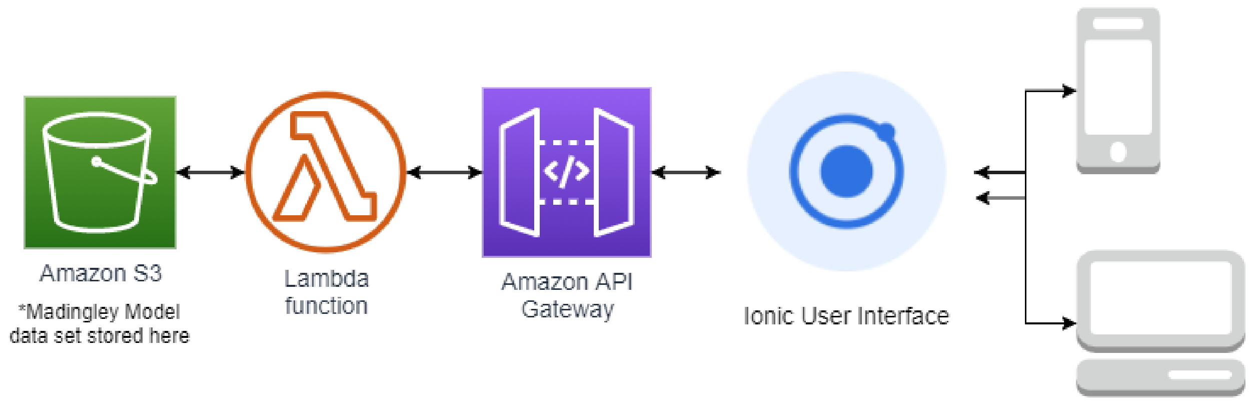


Figure 1: Architectural overview diagram

3.1 Application Components

3.1.1 AWS S3 Bucket

An Amazon S3 Bucket is a data storage service offered by AWS that basically allows for users to store data of almost any type in a file-based system within the AWS Cloud, which makes for easy and efficient management. For our application, we will be using an S3 bucket to store the Madingley Model dataset that will essentially be used for displaying different biological stats in a user's given area.

3.1.2 AWS Lambda Function(s)

The entire backend of our application will consist of AWS Lambda function(s). This is a serverless application that when prompted, will receive the data passed in by the user from the User Interface, and will use that received data to then pull our previously mentioned Madingley Model data from the S3 bucket, process it, and send it back to the User Interface to be further processed and displayed.

3.1.3 AWS API Gateway

The Amazon API Gateway is essentially what connects the front end to the backend. The previously mentioned Lambda Function(s) will be mounted to an API gateway endpoint. This API uses REST(ful) HTTP methods to pass the necessary variables from our UI to the designated endpoint, in the form of a JSON to be received by Lambda. Once the Lambda function processes this input and pulls necessary response data from the Madingley Model data set, a REST(ful) response is then formatted and all visualization data is sent back to the UI.

3.1.4 Ionic User Interface

The user interface being used for this application is Ionic. The Ionic framework uses universal web programming concepts to allow for one code base to be written which is then packaged into a progressive web application (PWA). This PWA format is able to be downloaded and run on mobile devices (both iOS and Android) and can also be hosted and run as a responsive web application from any device with a modern browser (Chrome, Safari, Edge, etc.).

3.2 Component Interactions

3.2.1 AWS S3 Bucket <---> AWS Lambda

After the Lambda function receives and interprets the data sent from the UI, it will then pull the corresponding data from the Madingley Model data set stored in the application's S3 bucket. This raw data will be in the form of a CSV, which will then be processed by the Lambda function to be packaged into a response and sent back to the UI via the API gateway.

3.2.2 AWS Lambda <---> API Gateway

After a request is sent via the Ionic UI, the API gateway will trigger an instance of our serverless Lambda function. The API gateway will pass the request data into the Lambda function to determine what Madingley data to be pulled from the S3 bucket. Once the lambda has pulled the desired scenario data from the S3 bucket, it will then return a REST(ful) response to the UI with that data in the body.

3.2.3 API Gateway <---> Ionic

Once the user has set all their desired variables/options, the Ionic application will use an imported angular HTTP client library to make a REST API call to the API gateway endpoint at which we've mounted our Lambda function to. This REST API call will be in the form of a POST method which is named so because it "posts" the JSON data that contains the desired user/location/scenario options to the body of the request. After the Lambda function handles the rest of the data pulling/processing, a REST HTTP response will be sent back to the front-end, and in the body of that response will be the data required to visualize the results of the imputed scenario.

3.3 Influences

Each of the components in our application's architecture has been chosen through intensive research and comparison, and is included because we feel it is truly the best candidate for its job. The Ionic UI was chosen because we believe it allows for a smooth Cross-platform development experience, with the ability to not only share a single code base, but to share even individual written components between iOS, Android, and Web application interfaces. This maximizes accessibility to the application, while still providing a smooth and easy development process. The AWS API gateway, Lambda, and S3 backend services were all chosen because Amazon has the highest regarded web services due to them being extremely easy to integrate not only with each other, but with a separate UI framework for a hybrid application like ours. They are also extremely reliable, and very easy to maintain especially for someone who doesn't have a computer science background. It isn't difficult for someone with no programming experience to learn how to use their intuitive services.

4.0 Module and Interface Descriptions

Our project will consist of five main modules: the back-end, the front-end, geolocation, multi-language support, and data exportation. The back-end module will process all the information received from the user. This module will have three submodules: AWS Lambda, AWS S3, and AWS API Gateway. AWS Lambda will take the information given by the user, process it, and securely return the appropriate results. AWS S3 will store all the Madingley model data for our application. AWS API Gateway will connect the front-end to the Lambda functions. Next, the front-end module includes all the components that the user interacts with. This consists of the Ionic interface and the output graphs. The geolocation part will retrieve the area of interest from the user. Data from these sections will be analyzed and converted to graphs. Additionally, we added multi-language support, so that our app could reach a larger number of users. Lastly, the data exportation module will allow users to easily store their output onto their own device.

4.1 Back-End

4.1.1 AWS Lambda

AWS Lambda is a serverless compute service that allows developers to run code without having to control or manage the physical hardware. It is a scalable compute node that can be grown or shrunk as need be.

For this project, AWS Lambda will be used as the primary back-end processing unit. This involves taking in requests made from the front-end ionic application, interpreting that request, then returning the appropriate results. While this seems simple, the complexity comes from the security measures put in place. The lambda function will be equipped with validity and field identification tools which will prevent malicious actors from gaining unnecessary access to this function, even if the gateway / middleman is compromised.

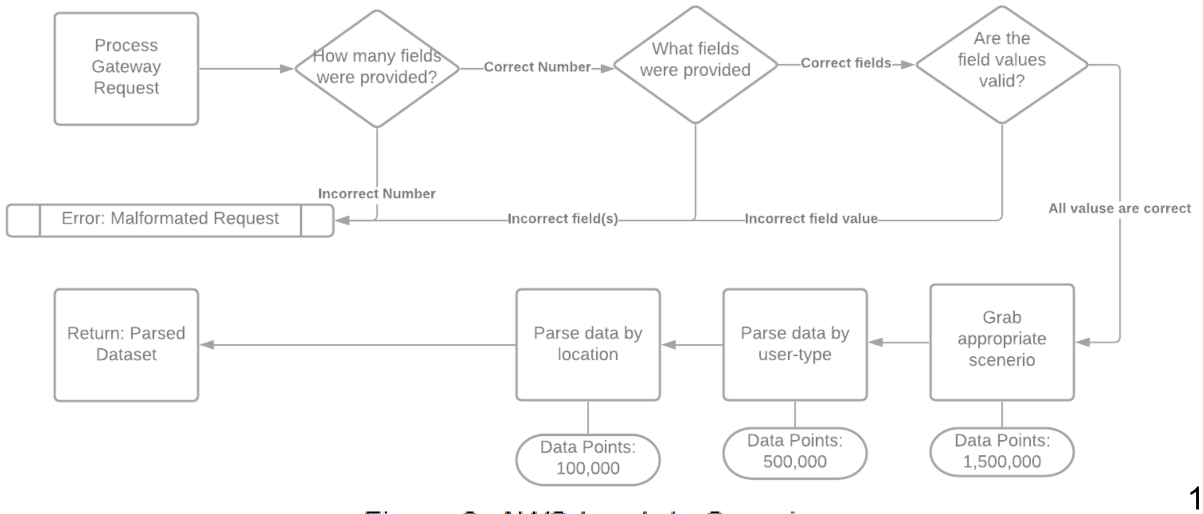


Figure 2: AWS Lambda Overview

Figure 2 outlines how the lambda function will prevent malicious actors from gaining insight into the function and association of the application. In layman terms, if the request is formatted incorrectly, in any way, it will not be processed. Figure 2 also showcases the control flow of how the lambda function will function and accomplish its purpose (input verification, data retrieval, data parsing). The exact parameters and their acceptable values/ranges associated with this function are still unclear since the scenarios, essential biodiversity variables (EBVs) per user-type, number of user-types, and radius types/sizes are still unknown. However, it is known that the following parameters will be requested: user-type, latitude, longitude, radius, and scenario.

4.1.2 AWS S3

AWS S3 is an abbreviation for Simple Storage Service. It is an object-based storage system that behaves and looks similar to most computer file storage systems. As such it yields a high level of usability for users who are unfamiliar with cloud-based storage solutions. Amazon engineers have designed the scalable environments for durability and as such claim 99.99999999999% of data durability. This is important for this project, since obtaining the scenario data is computationally expensive.
This component is responsible for maintaining the data, as well as version control of the bucket to prevent unnecessary or mistaken changes made to the storage environment.

Figure 3 indicates a basic file structure that could be used within the S3 bucket, with the parent folder being a given scenario, and the files within being associated with the geolocation data, and EBVs.

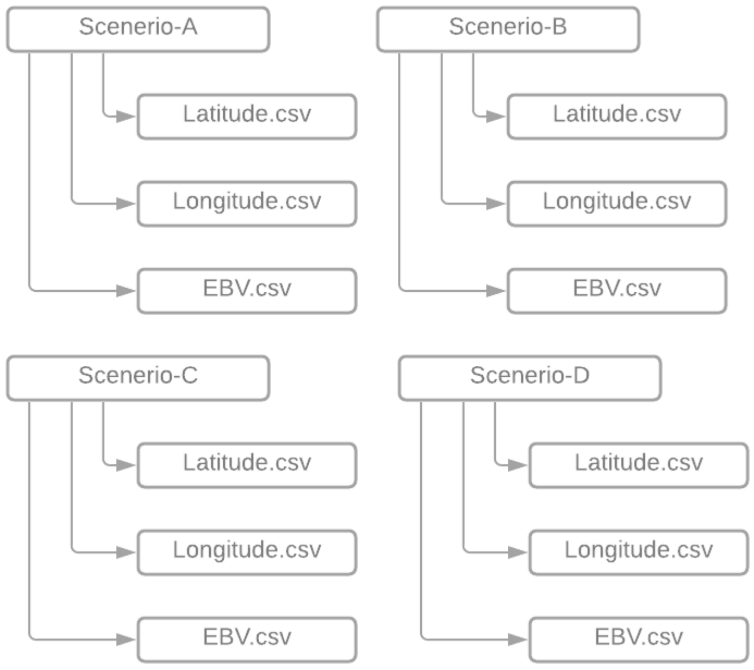


Figure 3: AWS S3 Overview

4.1.3 AWS API Gateway

Amazon's API Gateway is a full managed service that allows developers to create and maintain scalable APIs that can go hand-and-hand with other scalable services like: S3, and Lambda. The API Gateway is used to connect the front-end, ionic, to the back-end, AWS lambda. An API gateway is crucial since it acts as an obfuscation tool to prevent unauthorized parties from gaining insight into the inner workings of the backend. Secondly, this tool provides a developer friendly interface to connect the front-end components to the backend with little to no management after initial production.

4.2 Front-End

4.2.1 Ionic

Ionic is a mobile application software development kit built in TypeScript that allows for the development of progressive web applications that can run on a variety of device platforms. For development, Ionic allows the integration and usage of JavaScript frameworks that can be used to expand outside of the normal JavaScript functionality, these frameworks include:

- Angular
- React
- Vue

The Ionic SDK integrates all of the different components that our application uses, and also serves as the foundation for the base of our application using the JavaScript framework Angular. This includes the incorporation of our different software APIs and libraries, as well as manage the complexities of keeping the user interface simple. We have chosen this framework over the others due to the personal preference of the team based on the syntax and compatibilities Angular has.

Ionic produces progressive web applications, which are new and improved web applications that are able to support newer technologies and use modern techniques. One such technology is the ability to translate from one language to a variety of native device languages such as for iOS and Android. It is also able to support a variety of Javascript libraries for implementation, this is due to Ionic being written in TypeScript. TypeScript is a superset of JavaScript, and therefore allows the implementation of JavaScript frameworks, as well as JavaScript libraries, and nearly every functionality of JavaScript itself, due to TypeScript being a superset of JavaScript. Figure 4 depicts the hierarchy between all of the mentioned frameworks, libraries, and languages mentioned in this section.

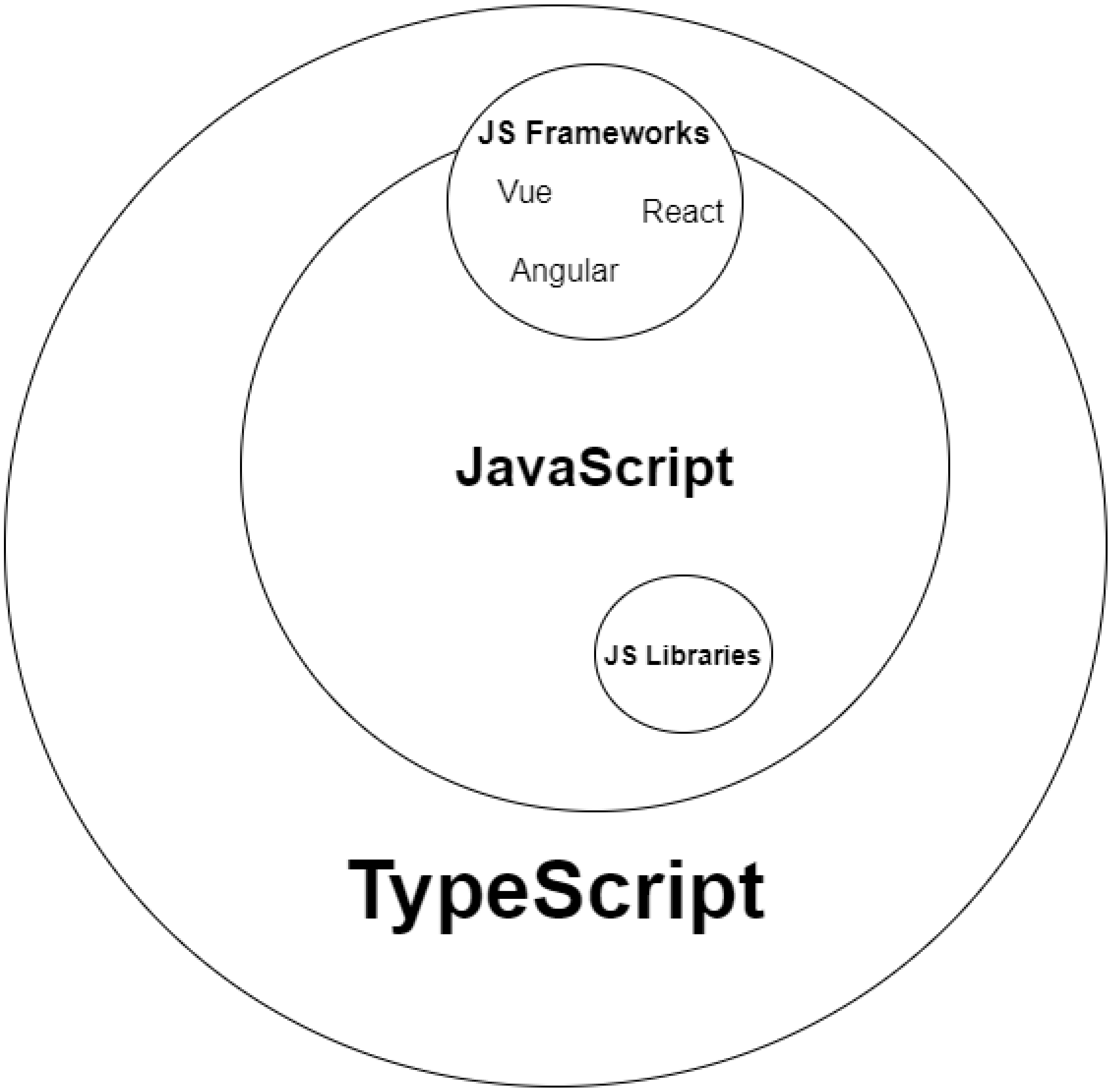


Figure 4: Hierarchy of Languages and Tools

4.2.2 Graphing & Visualization Libraries

Visualizing the data is one of the most important aspects for our end-users. Without the ability to visual the data generated from the Madingley model, the data has no significance to individuals that do not have a technical background on the subject. As a result of Ionic's popularity there are numerous libraries that we can use to visualize the data. The two primary candidates for data visualization are D3.js and Highcharts. D3.js stands for Data-Driven-Documents Javascript Library. It is a visualization library that displays GIS information i.e. latitude, longitude, and some metric. Highcharts is a Javascript library that allows developers to create interactive graphs and charts. The most important shared aspect of these two libraries is that they are both natively compatible with Ionic. Since they are compatible with Ionic, there won't be any performance slowdowns or bottlenecks as a result of these libraries. Figures 5 and 6 showcase basic examples of what D3.js and Highcharts are capable of doing.

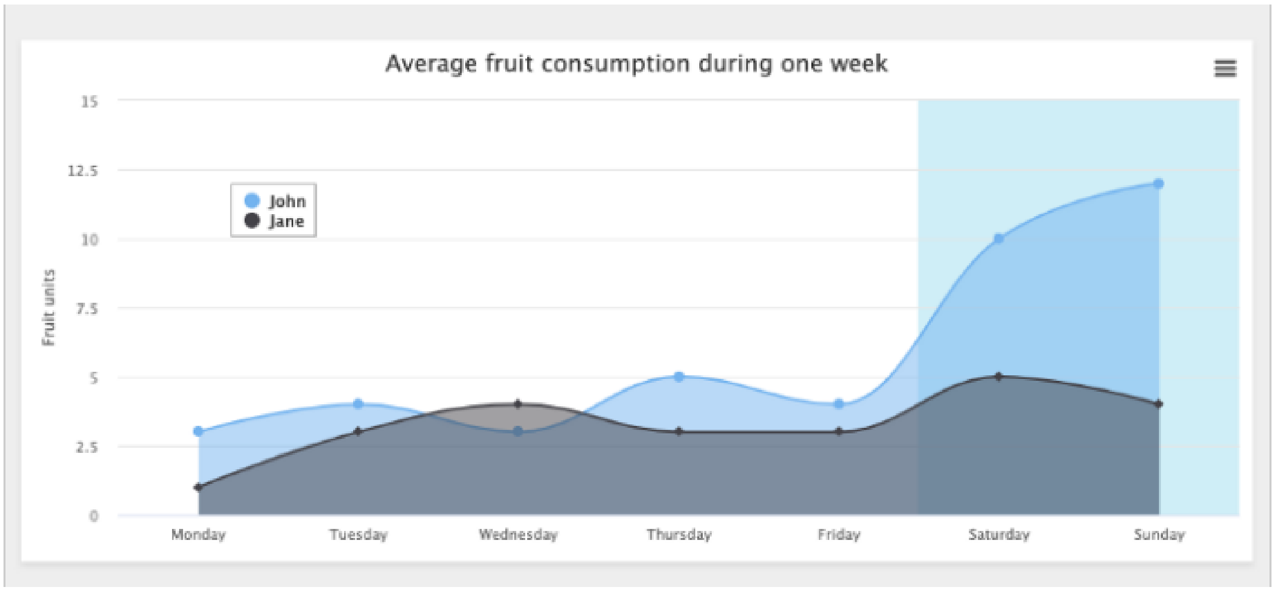
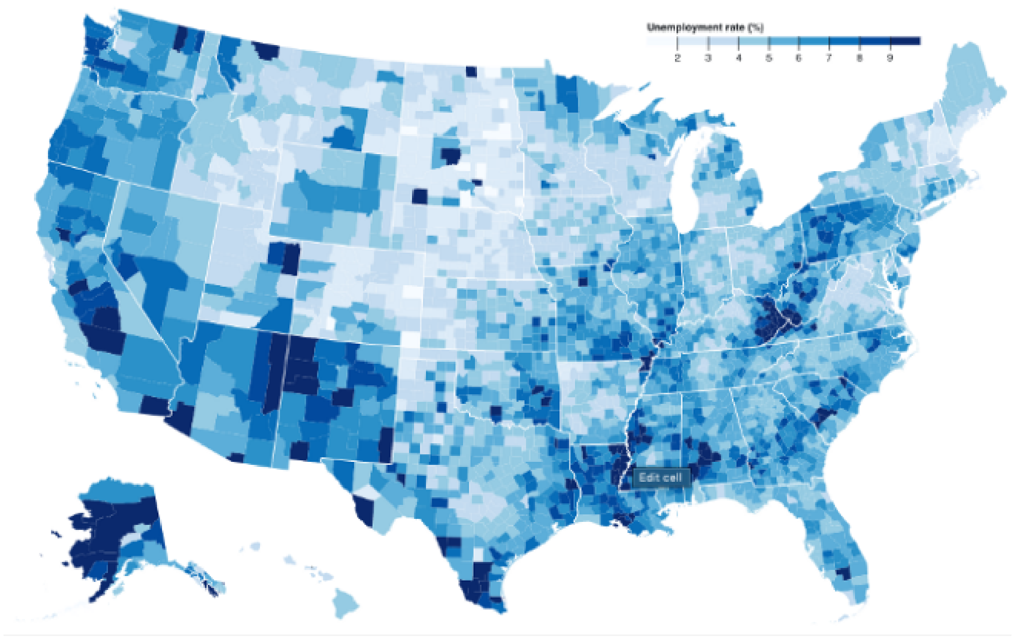


Figure 5: Highcharts visualization example



Data visualization over map using D3.js

Figure 6: D3 visualization example

In terms of input parameters, this module requires that information generated by the AWS back-end. This information includes, but is not limited to: user-type, chosen latitude and longitude, scenario-type, as well the requested data which will contain anywhere from three to ten columns of data. Two of the provided columns must contain geolocation information i.e. latitude and longitude, and the remaining columns will consist of the varying essential biodiversity variables (EBVs) as desired by the client, and user-type provided. An essential biodiversity variable is defined as a measurement required to study, and report on biodiversity and its change over time. It is unclear exactly what types of graphics will be generated as a result of the end-user's selection of various factors i.e. EBVs, user-type, and the time considerations of the data.

4.3 Geolocation

In addition to the front-end and back-end, another module is geolocation. The first step is the user deciding whether they want to use their current location or select an area on the map. In either case, we will then provide them with instructions on how to use the map features. Once they continue from the directions page, we will use the Google Maps Javascript API to display a dynamic map.

If the user chooses to manually select an area on the map, they will then be taken to a map with a box. This box will represent the area on which they want to analyze data. The user will be able to move the map until they get the exact area they want. They will then confirm the selected region. Once it is confirmed, the latitude and longitude will be sent to AWS S3, so the appropriate data can be retrieved from all the stored information.

There are two edge cases that we will consider for the "select location" option. The first problem we might encounter is the map zoom and selected area size. We calculated that a default box that is 400km by 400km will almost always include the desired amount of data. Should the user want to include a larger or smaller area, they can do this using the zoom feature built into the Google Maps UI. The only downside to letting the user select the area size is the amount of data that is present in that particular area. If there is too much data or too little data, we will prompt the user that they need to select another area. For example, if the user zooms in too much and the box becomes 5km by 5km, the probability of having enough data is very low. In that case we might have one or two sets of data, but it won't be enough to produce an adequate visual output. Therefore, the user will get a message similar to "please zoom out and select a wider area," and they will need to try again with a larger area.

The second edge case we might encounter is if the selected area doesn't have enough data even at the default zoom. This might happen at latitude values above 65 degrees N or below 65 degrees S. If this does occur, we will prompt the user that they need to select an area between 65 degrees S and 65 degrees N.

Should the user select to use their current location option, the location will be checked even before going to the next page. By doing it this way, the latitude and longitude don't need to be checked again. The current latitude and longitude can then just be stored for later use.

Compared to the selection part of the app, designing the "use my current location" option will be easier. However, there are two problems that might occur. The first problem that might occur is if the current location is incorrect. We will know this if the user denies the location given on the map. In that case, the user can either try again or go to the "select location" option instead. Another error that might occur is if the user denies app access to location services. In that case, we will prompt the user to allow the app to access location services. If the user never allows it, we will prompt them to select their location instead. Overall, this option will require the use of built-in functions. In Figure 7, you can see that we will also use 3 other functions: locationOption(), selectLocation(), and currentLocation().

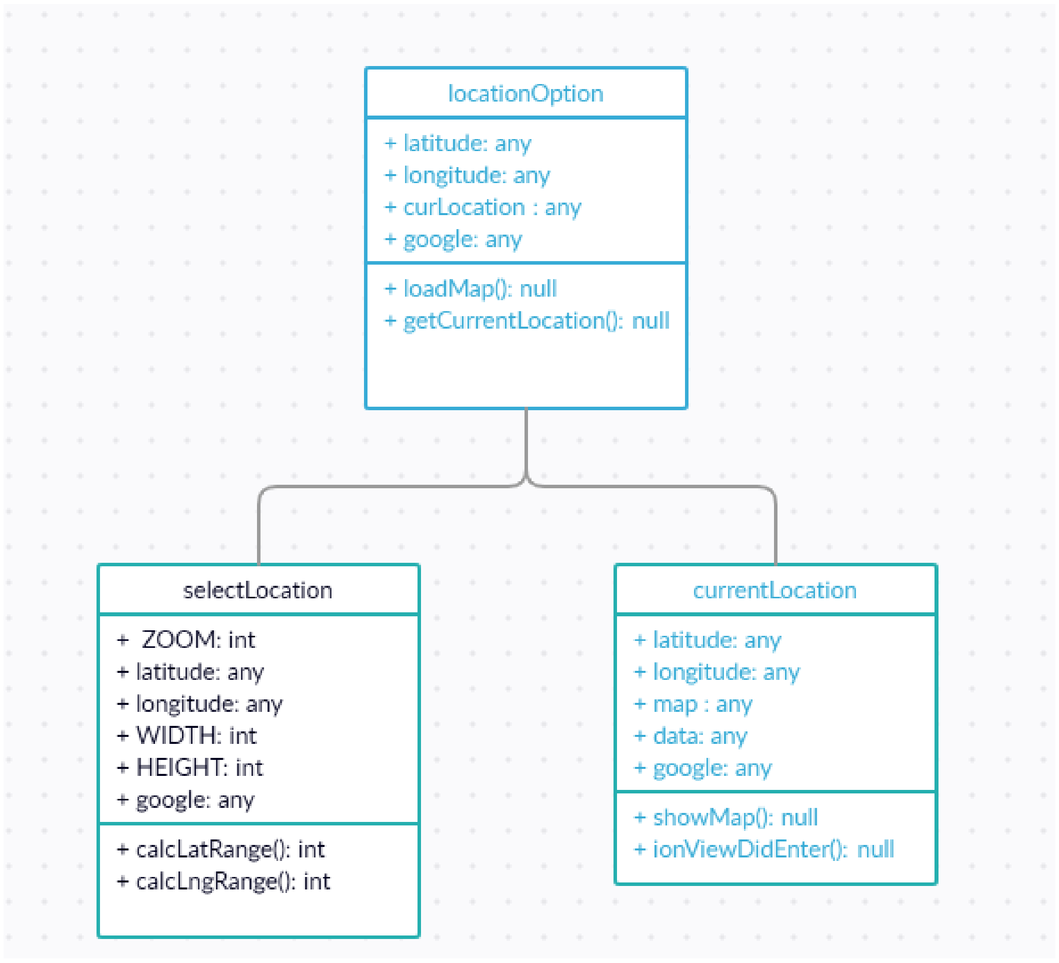
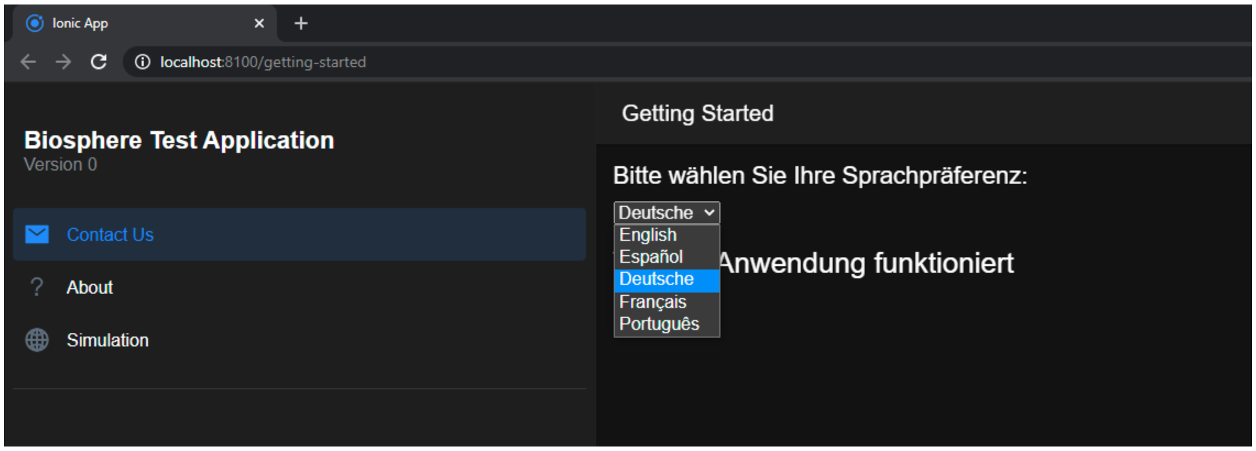


Figure 7: Geolocation Overview

]

4.4 Multi-Language Support

The multi-language support of the application is important to allow internationalization of our application. This is based on the idea that while interested users from overseas may not speak the same language as us, they may share the same passion or need for ecological science. Therefore, it is a high priority for us to make the application as easy as possible to navigate, and one such way we can do this is by supporting multiple languages. The implementation of this feature on the application is a list of languages that can be selected on the home screen in order to change the language of all of the text in the rest of the application. Figure 8 shows an example of a page in the Dutch language.



Bitte wählen Sie Ihre Sprachpräferenz:

Anwendung funktioniert

Figure 8: Multi-Language Support Example

This multi-language module primarily functions using ngx-translate, also known as the internationalization library for Angular. It is a typescript dependency that can be used in the Ionic framework with import statements and declarations of instances in the code. It is also installed using npm, a Javascript package manager used to simplify installation.

The documentation on this dependency is very thorough, with many third-party online resources available as well. The source repository also contains many details regarding implementation usage that will be beneficial when applied to our application. Specifically, it outlines how certain features and predefined functions can be used to do specific tasks. For example, one such function is getBrowserLang, which allows us to find out what the default language of the device is, regardless of the device type, and use it in any way necessary.

Other features that are being used from the library include the ability to store all of the data for one language in one single JSON file. Each language is represented by their language codes, e.g. “en” for English, “fr” for French, “de” for German, etc. Each JSON file contains sections designated for each specific page that gives out instructions or information that can be translated. These sections are then divided into different blocks representing each object in the HTML section of the page, allowing each block to be translated individually.

4.5 Data Exportation

The exportation of data is used for personally obtaining the data calculated and visuals displayed in the application. The usage of this feature will specifically allow the user to create a PDF of the data that has been displayed, such as: graphs, figures, and tables, then download the created file. An example of the creation and download functionality can be seen in Figure 9.

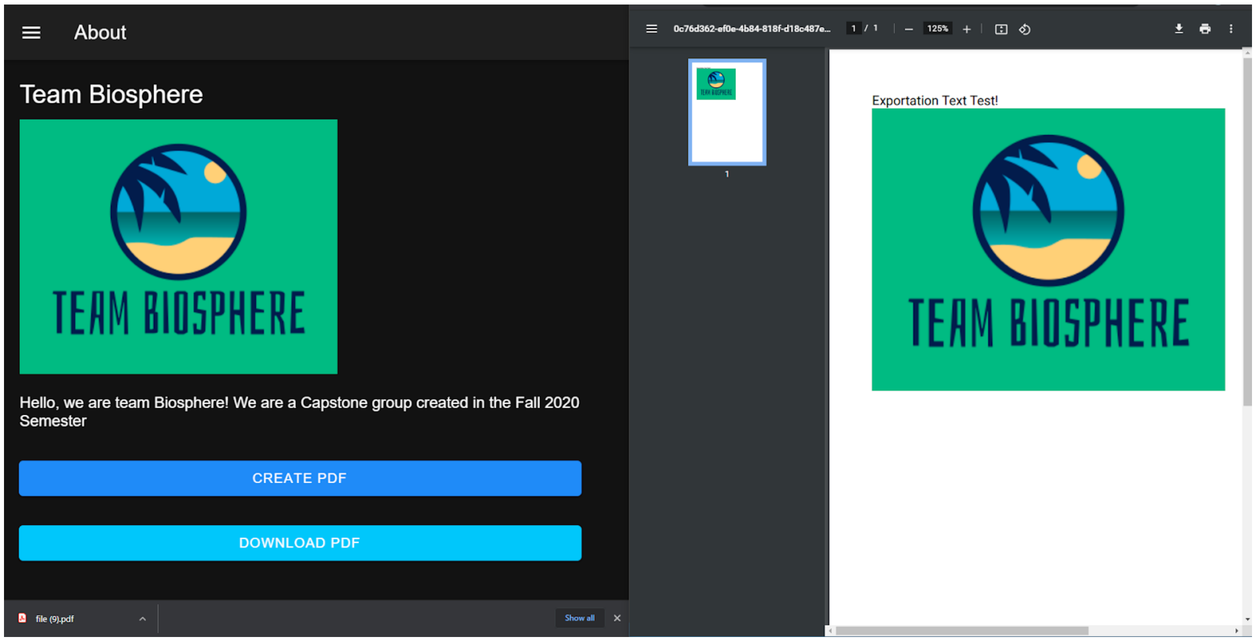

About
Team Biosphere
TEAM BIOSPHERE
Hello, we are team Biosphere! We are a Capstone group created in the Fall 2020 Semester
CREATE PDF
DOWNLOAD PDF
file (9).pdf
Show all
0c76d362-ef0e-4b84-818f-d18c487e...
1 / 1
125%
Exportation Text Test!
TEAM BIOSPHERE

Figure 9: Creation of PDF and Download Example

The module is mostly implemented using pdfMake, a document generation library that allows the application to create PDF files from data used in the application. After this PDF is created, the library also has a download feature that can easily be implemented into web clients with a few statements. However, the process for downloading and viewing PDF files on mobile clients is much more complex, which requires the usage of multiple other imports such as FilesystemDirectory and Filesystem from the Ionic capacitor core library. These two are used for accessing the mobile directories in the mobile clients which are much more precise and refined in comparison to web clients.

For general usage of the pdfMake library, the application creates a Blob object that holds all of the different details for the PDF file to be made. This object that contains the data itself, as well as the styling and fonts for the PDF; is able to be used with the built in createPdf function to assemble the file. This file can then be viewed using the object's open function or downloaded with the download function, both built in functions from the pdfMake library, with added steps for mobile clients.

Another notable aspect is that some data passed through to the PDF must be changed into specific data types, as an example, images must be converted to Base 64 Strings to be recognized as actual images in most cases. This can be done manually with a function that appends specific characters to a string, it can also be done using built in functionalities like canvas in HTML, although this may change in the future, and therefore a manual function is the preference of our application. This module will also be installed using npm, which allows for the entire pdfMake package to be installed into the list of node modules for the application.

5.0 Implementation Plan

In order to meet our final project deadline, the team has implemented a simple progression plan. This plan will keep us on track to deliver our final project on time. In the plan, each of the modules were split up into tasks. Furthermore, these tasks were then evenly distributed between the five members of our team.

Greg is focusing on getting the front-end design (graphical user interface) completed. Greg currently has the very basic layout completed. Going forward, he will add to this basic layout by adding backgrounds and other details to make the app easier to use. Greg's assignments correspond with the light blue parts of Figure 10.

Chufeng is working on the graphical visuals for our app. He will take the data selected by the user and translate it into graphs/figures. These figures will then be displayed to the user. In Figure 10, the light green task is assigned to Chufeng.

McKenna will be creating code that changes the language of the text in the app. The default language will be English. However, if a user selects a different language or their browser has a different default language, it will change the text to this language. The final product will be able to support a minimum of two languages: English and French. If McKenna finishes this task early, he will then work on exporting the user selected data. This is a stretch goal, so this will only be completed if all the other tasks are going smoothly. The purple-colored tasks in Figure 10, are McKenna's.

Kainoa is the main back-end developer for our project. Kainoa is currently finishing the creation of the AWS IAM roles. He is creating two different roles. One role is for developing the application and the other one is for practicing and experimenting. The next assignment Kainoa has is creating the AWS S3 environment for our project. He is also going to create the required lambda functions and an API gateway. In Figure 10, the dark blue boxes represent parts of the project that Kainoa is going to complete.

Wesley is working on the GPS features of our project. His current code allows the user to use their current location rather than selecting an area on the map. His next goal is to write code that allows the user to select a square grid of 400 by 400 kilometers. This grid will then relay the latitude and longitude points to the AWS server, so that we can retrieve the data pertaining to this location. Wesley will work with Kainoa to guarantee that the data retrieval process is streamlined. Wesley's geolocation tasks are represented by the gray color in the Gantt Chart.

The pink color in Figure 10, represents the two tasks that we are going to work on as a whole team. We are hoping to have the majority of tasks completed and working by the beginning to middle of March. This will allow the team to make sure that we have a really good alpha prototype. After the alpha prototype, all the tasks should be completed, and we will simply be testing and adding any last-minute details. This will ensure that most of the bugs are ironed out and that the app works the best it possibly can. On April 16th, we will be presenting at the UGRAD symposium. This will allow us to gain some excellent feedback on our project. We will use this feedback to pinpoint where to allocate our remaining time and where to do some of the final testing.

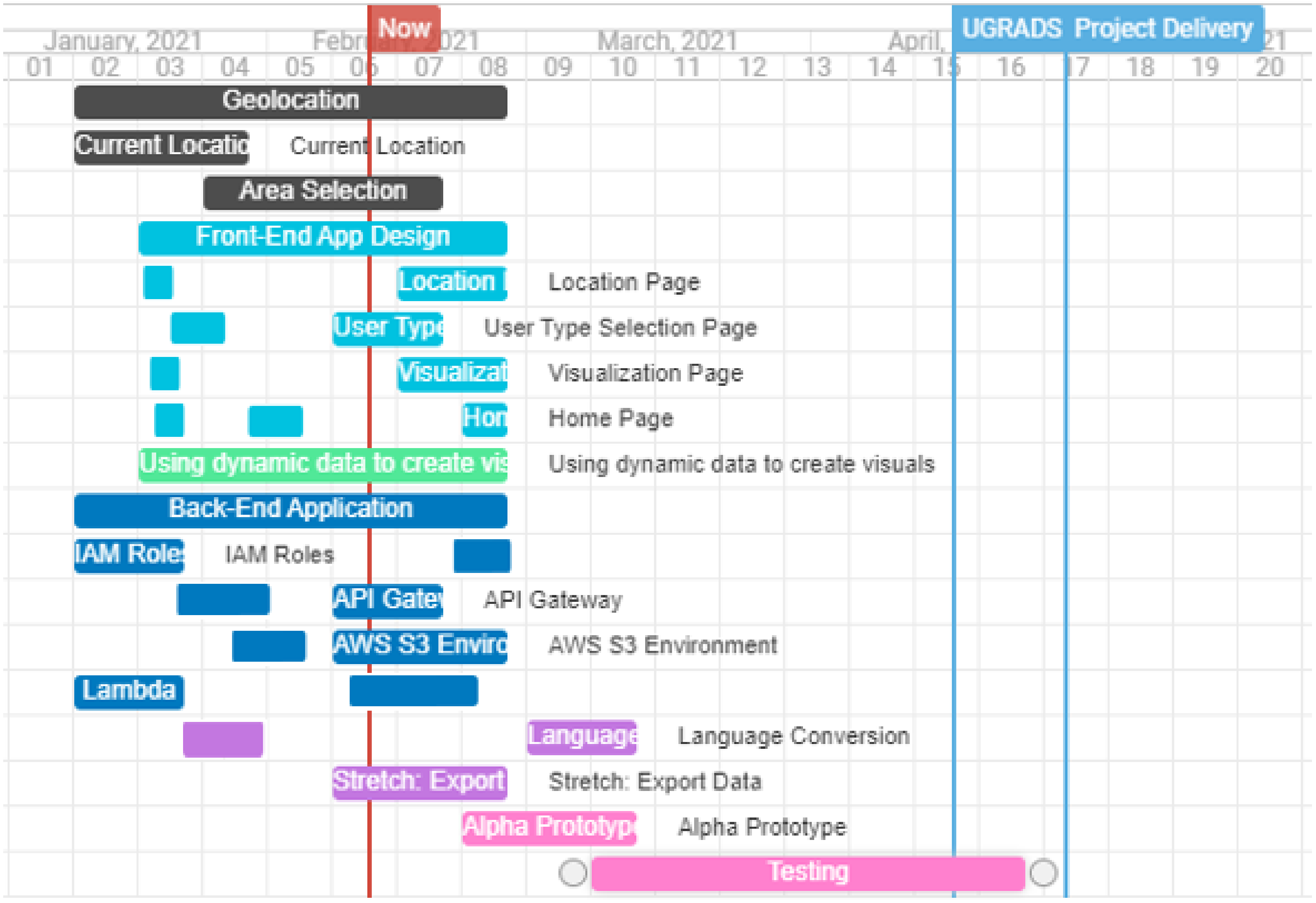


Figure 10: Gantt Chart showing the tasks that are to be completed by product delivery in April 2021. Each member has a corresponding color: Wesley (gray), Greg (light blue), Chufeng (light green), Kainoa (dark blue), McKenna (purple), team effort (pink).

6.0 Conclusion

Biodiversity and species richness are key measurements in ensuring that the ecosystems around the world are healthy. With a decline in biodiversity, there will be a negative effect on ecosystem services; for example air filtration, water purification, and pollination. Without these key activities, ecosystems will slowly dissipate into barren unlivable wastelands.

The goal of this application is to provide information to people, and organizations that care about the conservation of biodiversity and to slow the effects of climate change. This will be done by providing the target audience with access to the Madingley model. The model has the ability to predict essential biodiversity variables (EBVs) that are used to measure, study, and report the change in biodiversity over a period of time. Typically, understanding these variables and their overall impact is reserved for those who have existing technical knowledge on the subject. However, as a result of this application, the goal is to provide almost anyone with this information so that they can be informed global citizens and understand the true impact on some seemingly essential activities.

The intended audience includes policy makers, scientists, and the general public. More specifically individuals that care about biodiversity and species richness for various reasons from policy changes that positively impact biodiversity, to biodiversity research and management to a general user who wants to know about biodiversity and its future. This project's direct sponsor Chris Doughty, an associate professor of ecoinformatics, cares greatly about biodiversity and how certain activities have rapidly increased the rate of species lost.

The purpose of this document is to outline a more concrete plan regarding software selection, and design choices associated with the overall structure of this application. The components selected for this project were done so in the hopes that the largest number of individuals can be reached. These choices include constructing a PWA to not limit the target audience to a specific operating system or device, selecting AWS as the primary back-end management tool as a result of their data redundancy network, high-speeds, and broad reach around the world.

When the user interacts with this application the goal is to provide an easy-to-use solution, that is intuitive and informational for all types of users.