Technological Feasibility Analysis Report

October 23, 2020

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

Table of Contents

1 Introduction **3**

2 Technological Challenges **5**

3 Technological Analysis **7**

3.1 Application Platform 7

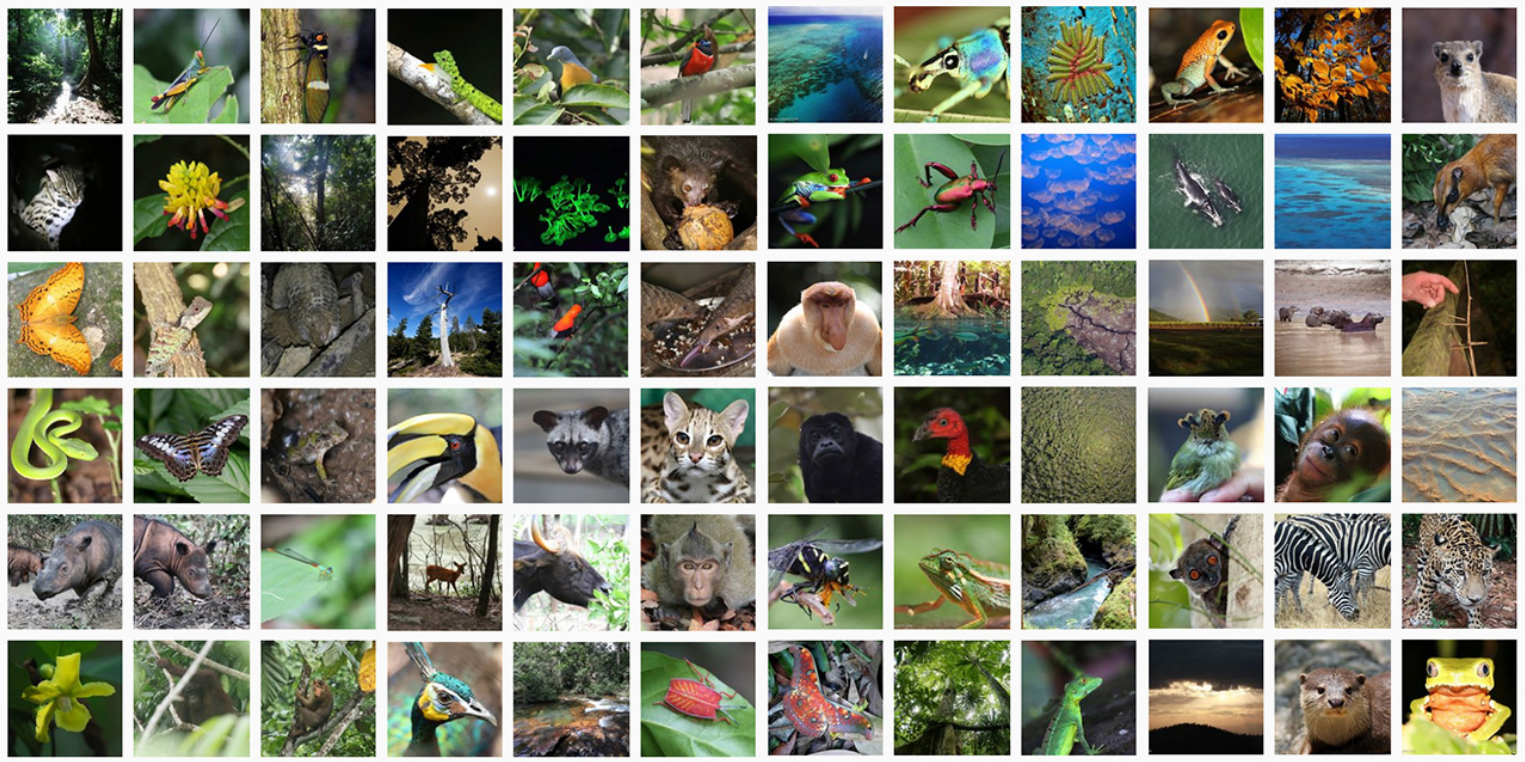
3.2 Framework 9

3.3 Storage 12

4 Technology Integration **15**

5 Conclusion **17**

Appendix. Summary of Major Design Decisions. **19**



1 Introduction

In the very near future, the northern white rhinoceros will exist only in memory, since they will become one of the many species that become extinct every year from the rapid increase of human intervention in the wilderness. The northern white rhinoceros in particular, has had it's population drastically reduced down to just two females in recent years, due to an immense amount of illegal poaching for their horns. Poaching however, is not the only problem that plagues the biodiversity of places like tropical rainforests, many other human related factors like climate change and logging cause drastic changes in the diversity of life. Therefore, it is imperative that people everywhere be forewarned about the repercussions that stem from the destruction and plundering of these luscious tropical habitats in biodiverse areas.

To help assist with this ongoing issue we, Team Biosphere, are working on creating an application to spread awareness across different platforms. This application solely aims to spread awareness about the preservation of biodiversity in tropical forest regions that are vulnerable to human industrialization and interference.

Our team sponsors, Chris Doughty (primary client) and Andrew Abraham (mentor) have spent numerous years investigating the interactions between forests, climate, and how animals affect ecosystem functions. They have commissioned various projects in the past to create an application to fulfil similar tasks to ours. However, these other applications fail to fully grasp the potential of how versatile and adaptive modern technology can be.

Even though modern technology has developed the capability to access data across different platforms, previous projects on this subject still fail to use this feature. These applications limit their users to a specific format, such as exclusively to either mobile or web users. To overcome these limitations, Team Biosphere, plans to utilize a new technology known as a progressive web application (PWA), that can be accessed across any mobile or web format.

The application that Team Biosphere presents aims to support a wide variety of devices and avoid limiting the client's intended audience to one specific platform while providing the same high quality data and information across all mediums equally. This includes access from any device that is able to use an internet browser, from computers, to both Android and iOS devices. A broad spectrum of choices for the users ensures that the application does not limit the target audience. If the target audience is limited, our ability to spread awareness about the loss of biodiversity is negatively impacted.

The application itself will have the ability to visualize the data generated from the user defined or predefined scenarios. A scenario is a sequence of predicted / theoretical events that take place over a period of change. The primary goal for this application is to select a specific location using an interactive map, or with permission, using the user's location. Using this location information, the application will display information about the biodiversity in the area. The user will then have the option to choose potential effect scenarios including: deforestation,

climate change, and poaching. These scenarios will be used to visualize the effects of an action on an ecosystem and its levels of biodiversity.

Since storing and displaying the exact data on every single species on Earth is extremely impractical and nearly impossible for ordinary devices, the data displayed is derived from the Madingley Model. This scientific and peer-reviewed model provides a way to feasibly represent life in ecosystems by using the unique interactions between animals to aggregate their data into units called cohorts. In the technical sense, these cohorts occupy a much smaller amount of digital space, and allows the model to be less computationally expensive.

In the model, the cohorts are able to accurately represent ecosystems by using the collected data of any given biome or region including:

- ❏ The interactions between species such as predators and prey
- ❏ The total number of individuals
- ❏ The biomass content of both plant and animal life

These factors are then processed through a series of complex calculations made specifically for the Madingley Model to define cohorts. They are also able to add elements to the formulas to estimate how futuristic scenarios would affect the environment. One such impact would be a trophic cascade, where an important species goes extinct and causes a huge change in the food chain, which in turn changes the cohorts in a selected area. Team Biosphere's application intends to use these calculated values to display to the user and show specifically how the scenario selected affects the chosen area.

The need to develop this application is the reason why Team Biosphere was established by undergraduate computer science major students at Northern Arizona University. The team includes students: Kainoa Boyce, McKenna Chun, Gregory Geary, Wesley Smythe, and Chufeng Zhou, as well as the team's mentor: Andrew Abraham. This document will provide an overview of the different technical aspects of our project and our options for dealing with each particular problem. It will also include the chosen candidate for each challenge, which will be researched more thoroughly and used throughout the development process.

2 Technological Challenges

The focus of this section is to outline all foreseeable technical challenges that could occur during the development and system design process. The system will be composed of the following components:

- ❏ **Code/Application Repository**: A central location where all code and documentation will be held.
- ❏ **MVP Back-end**: A program used to generate, process, and visualize the request generated from the minimum viable product (MVP.)
- ❏ **Stretch Back-end**: A program used to allow end-users to define their own scenarios. This data is then passed to the MVP Back-end for visualization.
- ❏ **Storage**: An cloud-based storage system used to house the data associated with this application.
- ❏ **Web Variation**: A browser based variation of the mobile application.
- ❏ **Mobile Variation**: A mobile application.

Each of these components live in one of three domains. The domains are used to group components by their function and user access. Each domain and their respective components are defined as follows:

- ❏ **Client**: This domain is accessible by the client and consists of all front-end / public facing components.
  - ❏ Web Variation
  - ❏ Mobile Variation
- ❏ **Server**: This domain is accessible by application administrators.
  - ❏ MVP Back-end
  - ❏ Stretch Back-end
  - ❏ Storage
- ❏ **Admin / Developer:** This domain is only accessible by application developers and administrators.
  - ❏ Code Repository

2.1 Cross platform development

The client has stated that it is ideal to have an application that is functional on both mobile and desktop devices. In order to satisfy that requirement a single application must be compatible across desktop and mobile machines; or multiple applications must be created, one per device, and or operating system. The challenges associated with cross platform include:

- ❏ unpredictable display complications due to display screen variation.
- ❏ a lack of component compatibility due to a non-native approach, or multi-code-base approach.
- ❏ lack of support for specific operating system or browser versions.

2.2 Data Storage

The scenarios associated with the Madingley Model are

The data associated with each scenario range from 0.2GB to 8GB. As a result of the large file size, a data storage component is necessary to stash these files. According to our client the peak capacity will be about 1000GB. Given the possible max capacity of the file system, this system must be easy to access, put data, and fetch data at high-speeds regardless of its current capacity. In order to achieve this the storage system must be located in multiple locations at once; this feature also allows for data redundancies to prevent data loss. The primary concerns associated with data storage include:

- ❏ Cost per query or cost per gigabyte.
- ❏ Data availability.
- ❏ Compatibility with other components.
- ❏ Read and write speeds.

2.3 Back-End Data Processing

Given that, the Madingley Model is computationally expensive, it is unrealistic to expect the end-user to have the hardware to process the raw data generated from a scenario (*See 2.2 Data Storage*.) As such, a back-end system is required to process the information. The primary options for back-end systems include: privately owned hardware or scalable cloud back-end systems. Our client does not have the budget or space to accommodate a dedicated server therefore a cloud system must be used. The primary concerns associated with cloud systems include:

- ❏ Cost per activation.
- ❏ Unfamiliarity with serverless functions.
- ❏ Cloud system integration.
- ❏ Cloud system management.

2.4 Security

This application and system design require access and management of numerous resources (i.e. data storage, application hosting, cloud service integration and management.) This allows for multiple attack vectors. In order to prevent malicious actors from gaining unfettered access into these systems, the must design the application and its components with security as the primary focus. Security as a development forefront means limiting the user and system access to a bare minimum, this is known as the principle of least privilege. By following information security best practices the application will not only be more secure but have a longer longevity because its design will not be obsoleted by Common Vulnerability and Exposures (CVEs.)

3 Technological Analysis

The main focus of this section is to provide an in-depth analysis of the major design decisions that are required in order to establish a foundation on which to begin the project development. This section consists of an overview of the more realistic choices, a further evaluation of those choices, and a justified conclusion for each final decision. The four major design decisions being evaluated are:

- ❏ Application
- ❏ Framework
- ❏ Storage
- ❏ Data Processing & Visualization (back-end)

3.1 Application Platform

The application must be available to use on one or more different platforms. With this being said, and given that our potential end user's of this application already own either an iOS, Android, PC, or combination of those, devices, the top prospects for the type of application are as follows:

- ❏ Native Mobile Application
- ❏ Web Application
- ❏ Progressive Web Application

3.1.1 Potential Application Platforms

In order to determine the best application type, below is a qualitative comparison of the advantages and disadvantages of each application type.

Native Mobile Application(s): A native mobile application is an application that is developed specifically to be used on a single mobile platform.	
Advantages	Disadvantages
1. Native applications have access to more tools and components.	1. In order to meet the client's specifications this would require at least two different code-bases.
2. Native applications typically have better performance.	2. The interface would only be accessible via mobile devices / platforms.
3. Testing native applications is easier than non-native applications, since native application are made for a single platform i.e. Android or iOS.	3. Requires the user to download and install the application, rather than running it off a web-server.

Table 1: The positive and negative aspects of using a native mobile application for this project.

Web Application: A web application is an application that is developed to be used by any device with an internet connection and access to a web browser.	
Advantages	Disadvantages
1. Not limited to a specified device or operating system.	1. Requires an active internet connection / unavailable offline.
2. Much easier development process, testing only needs to be done on different browsers and screen sizes	2. Slower than mobile apps, and less advanced in terms of features.

Table 2: The positive and negative aspects of using a web application for this project.

Progressive Web Application (PWA): A progressive web application is an application delivered through the web, built using web technologies (i.e. HTML, CSS, JS.) It is intended to work on any platform that uses a browser, including both desktop and mobile devices.	
Advantages	Disadvantages
1. Available to any device with a browser.	1. Difficult development process since testing must be done on all 3 platforms (iOS app, Android app, PC web browsers)
2. Can be downloaded as a native mobile application for both iOS and Android devices.	2. Our team has the least experience with PWA development

Table 3: The positive and negative aspects of using a progressive web application for this project.

3.1.2 Chosen Approach

The choice we decided to go with was the Progressive Web Application. The issue with developing two separate native mobile applications is that it entails far too much overhead and still doesn't provide a web application. The issue with just doing the web application is that there would then be no offline version of the application. It was determined that the target audience prefers an application interface in comparison to a web browser for mobile devices. The only downside to developing a PWA is that the team has very little to no experience with that realm of development. The team plans to overcome this because the development of the application as a PWA allows for the highest possibility of widespread use, and provides the most accessibility to the application for its potential users.

3.1.3 Proving Feasibility

In order to prove that a progressive web application is the best choice for this project. The team will develop sample applications that will run across multiple platforms. The team will demonstrate the increased usability for users in both the mobile and web applications.

3.2 Framework

Given that application being developed will be a PWA, this section outlines and evaluates various possible frameworks. The possible choices for this design choice has been narrowed down to the following:

- ❏ AWS Amplify's built-in UI framework
- ❏ Ionic
- ❏ React
- ❏ Flutter

3.2.1 Potential Frameworks

In order to determine the best framework for this project, below is a qualitative comparison of the advantages and disadvantages of each framework option.

AWS Amplify's built-in UI framework: AWS Amplify is a cross-platform development framework belonging to the mass cluster of Amazon's web services. It was designed to allow for extreme ease of integration, however being a newer technology, it still has many kinks that may need to be worked out.	
Advantages	Disadvantages
1. Consolidation of tools required for application development (AWS Console.)	1. Can be sporadic, somethings work sometimes and not others for seemingly no apparent reason.
2. Smooth integration with other AWS services.	2. Newer technology and has far less online resources for trouble-shooting/learning.

Table 4: The positive and negative aspects of using AWS Amplify's built-in UI for this project.

Ionic: Ionic is another development framework used for building cross-platform mobile applications, and web applications. The framework uses a shared library of web development standards which allows for great PWA development.	
Advantages	Disadvantages
1. Many built in UI features within the framework.	1. Deeper customization to native devices is much harder to achieve.
2. Runs the most consistently throughout all desired platforms.	2. Possible lack of access to native Components.
3. Development process is easier because of this consistency.	

Table 5: The positive and negative aspects of using Ionic for this project.

React Native: is a cross platform development framework created by facebook that has become popular amongst developer's. Because of its ability to use javascript code to essentially directly control the native platform UI, React Native has become a favorite among those who are looking for great performance on native platforms.	
Advantages	Disadvantages
1. React has a lot of Native features embedded in its code base.	1. Code base doesn't transition as easily between platforms.
2. Uses existing and known web languages.	2. React Native does not come with a lot of pre-built are require conditional logic for multiple OS.
3. Simple development process	

Table 6: The positive and negative aspects of using React Native for this project.

Flutter: Flutter is Google's new cross platform development framework that offers many built-in widgets and tools that compile on the native platform itself incorporating all major platform differences minimizing the amount of shared code within the application.	
Advantages	Disadvantages
1. Newer possibly "up and coming technology" for cross platform development. 2. Offers great mobile application performance	1. Uses a new language, Dart, which does not have a large community.

Table 7: The positive and negative aspects of using Flutter for this project.

3.2.2 Chosen Approach

The team decided to go with Ionic as the framework choice. AWS Amplify was a favorite for a short time because of the consolidation of resources that would be used with the UI and Backend both being AWS, along with the expected ease of integration from front to backend, however the newer and somewhat sporadic framework was thought to cause unnecessary difficulty to the team, given the developers have little to no cross-platform development experience. React was a solid contendant as well since it is a fast growing skill among developers and offers great mobile performance, the only issue was when researching and testing, the team found that the amount of code that React required much more code to develop. Ionic, on the other hand, does not have this unnecessary overhead. Flutter being the shiny new framework that it is, was an intriguing choice especially because it might become a very popular framework in the near future, however when conducting creating sample applications, it was determined that Flutter was too difficult to work with especially because of the fact it uses a separate programming language Dart, rather than being primarily composed of Typescript/Javascript. Because Ionic was found to provide the most consistency between the three platforms, along with its fair share of online resources, offering a much smoother PWA development experience, it was the final choice for the application's framework.

3.2.3 Proving Feasibility

In order to prove the feasibility of Ionic as a framework for developing a progressive web app. The team plans to build a sample PWA using Ionic, and documenting its ease of use between mobile platforms and browsers. This will demonstrate that this framework will provide for the most efficient development process while also meeting all the requirements necessary for a fully-functional PWA.

3.3 Storage

Data storage will be key in providing data in the progressive web application, and we are given multiple choices in regards to how we will store the data. As this is a progressive web application, we are able to adapt and utilize some forms of normal web application storage tools in our development. Since one of our major challenges is specifically the accessibility to an extremely large quantity of data, it is essential that our application has a reliable and fast database that can fetch data quickly and efficiently for our users. We also require a low level API for local data management in our progressive web application, the selection that we are given extends to all current available normal web application APIs.

3.3.1 Potential Storage Options

In order to determine the best storage option, below is a qualitative comparison of the advantages and disadvantages of each storage option.

Amazon Web Services S3: Amazon Web Services or more commonly known as AWS, offers a wide range of tools and services that have been refined to assist in all sorts of situations for development. In our case we are looking into AWS's S3 module that allows for the easy access of data chunks that can vastly range in size, allowing from a single byte all the way to a few terabytes, and can all be transferred. It does this by using it's own developed object called a bucket, which allows any sort of data being transferred to be stored in a bucket object for easy sending and retrieval of the data.	
Advantages	Disadvantages
1. Official documentation on Github on support between AWS and the Ionic Framework allowing for easier development.	1. Possibly costly in the long-term with hidden fees
2. Fast executions of retrievals of data, extremely useful for our application that needs to move data to users quickly.	
3. Can transfer extreme amounts of data at a time, ranging from bytes to a few terabytes.	

Table 8: The positive and negative aspects of using Amazon Web Services S3 for this project.

Microsoft Azure Blob: Microsoft Azure is AWS's main competitor in the field, as Azure itself is extremely similar to AWS in terms of having many different database services for specific uses. The rival to AWS's S3 module is Azure's Blob service, which offers a similar experience in having its own object that it uses to transfer data. However, instead of a bucket object, Azure uses typical data types such as tables and Binary Large OBjects, commonly abbreviated as BLOBs. While it does not offer the peak performance as other options, this is very budget friendly and scales well towards smaller projects such as ours.	
Advantages	Disadvantages
1. One of the most economically viable options for storage space.	1. While extremely reliable, can be somewhat limited in terms of usage.
2. Reliable and trustworthy enough to transfer data securely and efficiently on the scale of our project.	
3. Can scale the amount of storage space for project sizes, from large multi billion dollar company programs to small group projects.	

Table 9: The positive and negative aspects of using Microsoft Azure Blob for this project.

Google Cloud: Google Cloud is one of the most popular cloud storage providers for consumers, rivaling other tech giants Microsoft and Amazon in terms of pricing and capabilities. It also shines in accessibility from anywhere on the globe, where accessing data is fast and easy due to low latency provided from it's numerous hosting locations that can store and send data.	
Advantages	Disadvantages
1. Can be cheaper than AWS in specific instances, depending on what services are chosen.	1. While extremely reliable, can be somewhat limited in terms of usage.
2. Allows for fast and easy accessibility from anywhere in the world.	2. Very simplistic in terms of storage and tools, does not provide much outside support other than the basics.
3. Does not have a minimum size for objects being transferred, can select as much or as little data requested.	3. Many instances it becomes more costly than AWS and may offer lower quality service.

Table 10: The positive and negative aspects of using Google Cloud for this project.

Kumulos: A smaller company than the three tech giants, Kumulos provides storage space for numerous prominent companies that can prove it's a viable option in comparison to their larger rivals. Kumulos is slightly different than the other options in here as well in technical terms, as it not only allows consumers to get storage space but also provides a wide range of services to help develop mobile and web applications.	
Advantages	Disadvantages
1. Helps to report crashes that occur when transfering data.	1. Mostly aimed towards standard mobile application development.
2. Has many smaller features included in it's own SDK that can help to develop and fix problems.	2. Less known than the other options, so less community development and support.

Table 11: The positive and negative aspects of using Kumulos for this project.

3.3.2 Chosen Approach

It was determined that AWS S3 will be the best solution for the application. AWS S3 provides numerous helpful tools in regards to developing a database and maintaining it. It also provides top quality service with a 99.9% uptime as well as quick and easy data transfer.

3.3.3 Proving Feasibility

The capabilities of AWS S3 can be demonstrated by creating a moderate sized sample dataset or data lake using their service and running numerous tests transferring data back and forth from our application to the cloud storage. This test could be done and tried with many different factors, a few including: location from where the data would be accessed. Since our application will be used around the world it is important that AWS S3 has numerous data redundancies to allow users to retrieve data around the world, and at any time.

4 Technology Integration

One of the main challenges so far was finding a way to integrate several different services that could satisfy all the requirements. We needed to find services that could add a UI framework, data storage, data processing, and visualization component. Finding services that work together for that many different things is difficult, but it is possible. Fortunately, one of the services we were looking at allowed us to add multiple components with just one service. We believe that the combination of Ionic and AWS will allow us to build a progressive web app that implements all the requirements.

The first part of our solution will be the data storage through AWS. This will allow programmers to store the predefined simulations in a particular location, so that they can be accessed by the user later. When the user selects the simulation, it will retrieve that data from the AWS storage. From there, it will use the Ionic defined front end to display the information to the user.

Another part of the project that will require a couple of the services is the UI framework. Once the user enters an area that they want information on, the information will be retrieved from the proper spot in the data storage. This might be kind of difficult depending on the region that was selected. If a user selects a wide range of biodiversity, we will be tasked with getting cohort information from several different spots of memory storage. Parsing through the data stored will require an algorithm tailored specifically to the Madingley model.

Additionally, if we use Ionic it will allow us to create an iPhone application, Android application, and a web application all at the same time. This will save us time from having to learn Swift programming language or writing two different sets of code for web applications and an Android application. This will greatly reduce the amount of time that is needed to implement the UI interface across multiple devices.

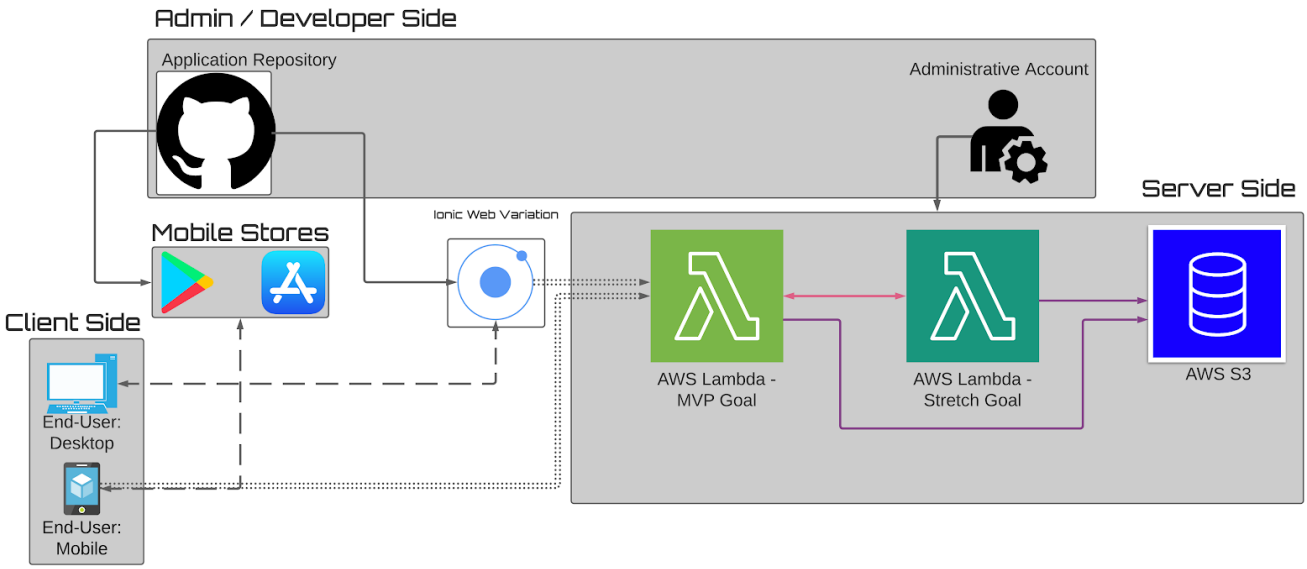


Figure 1.1 These are the key components of the application. There are three sides to the system design: admin/developer side, server side, and client side. To connect each component in a side and connect one side with another side, our application uses five different interactions. The first interaction that is shown in this diagram is represented by a solid black line. These connections can only be accessed by privileged users or developers. The second line style is the dashed black lines. The dashed lines symbolize an end-user who makes a request to a public system. For this program, the public system will either be a mobile application or an Ionic Web application, depending on what the user selects. The third connection is the double-dotted lines. This line formation is used to show how the end-user indirectly accesses that part of the system. Another interaction we see is the solid pink line. This line models the relationship between the MVP and the stretch goals. This relationship is necessary because the MVP will be used to process the stretch goal data. Lastly, the solid purple line demonstrates the connection between the Back-end function and the database.

5 Conclusion

While it is sadly much too late to save the northern white rhinoceros, the sorrow from this loss of biological diversity will not have to repeat itself in the future. With this application, we expect that our users around the world will become more aware about the saddening and catastrophic scenarios that may occur from continuing malicious human activities in the wild.

We also hope that the project is able to gain enough support in order to finally bring about some change in this long standing issue that has lurked on our planet for so many years. Though this may seem like a simple optimistic thought, with the application being able to support such a wide variety of devices, the diversity of our users will hopefully be as diverse as the habitats they will learn about.

During the design, development, and implementation of this project it is predicted that we will encounter numerous issues regarding: security, cross-platform compatibility, back-end data processing, and data storage. The above-mentioned concerns were addressed and have been outlined in a general plan including the technology of choice, and alternatives.

The table below (*see Table 12*) outlines a high level summary of the forecasted design challenges as well and the outcome.

In terms of the platform choices, we chose a Progressive Web application because it would allow for the development of a mobile and web application using a single code base. The greatest concern with this decision is that the team has little to no experience developing PWA's.

Regarding the framework choices, the initial candidates include: AWS Amplify's built-in UI Framework, Ionic, React, Flutter. It was determined that Ionice would be the best choice because it provided the most consistency, in comparison to the other candidates. Ionic also has a healthy online ecosystem, allowing a wide array of online resources, which will offer a much smoother PWA development experience.

The primary challenge associated with the storage aspect involves the large file sizes. The application must have a reliable and fast storage system to quickly fetch data for the end-user. As a result of AWS S3's storage options, ease of access, and reliability it was the best choice for development.

Challenges Summary		
Challenge	Choice	Choice Description
Application Type	Progressive Web Application (PWA)	A PWA will allow us to develop for a website and mobile app using a single codebase.
Framework	Ionic	Ionic is an established PWA framework that comes with numerous built-in components. Secondly, Ionic has a rather gentle learning curve, since it's a javascript based framework.
Storage	AWS S3	AWS S3 offers a cheap, fast, and user friendly way to store large sums of data.

Table 12: Challenges Summary Table

Appendix. Summary of Major Design Decisions.

	Conclusion Table											
	Platform development				UI				Data Storage			
Possible probelms	unforeseen display bugs due to screen size availability lack of component compatibility due to the non-native approach lack of support for specific OS versions				Adaptation of different screen models				The large amount of data makes it difficult for users to query data on the mobile side			
		Pros	Cons	Choice		Pros	Cons	Choice		Pros	Cons	Choice
Alternatives	Native Mobile Application	1. Tools and components aren't limited to that of a cross-platform framework 2. Applications usually run smoother with fewer bugs 3. Testing is only required to be done on a single platform platform The ability for user to have an offline version of the application	1. We would need a native app for iOS, and a separate native app for Android in order to provide an application available for widespread use, resulting in an enormous amount of overhead. 2. The interface would only be accessible via a mobile device		AWS	1. Smooth integration with other AWS services 2. Consolidation of tools required for application development (AWS Console)	1. Can be sporadic, somethings work sometimes and not others for seemingly no apparent reason. 2. Newer technology and has far less online resources for trouble-shooting/learning		Amazon Web Services	1. Official documentation on Github on support between AWS and the Ionic Framework allowing for easier development. 2. Fast executions of retrievals of data, extremely useful for our application that needs to move data to users quickly 3. Can transfer extreme amounts of data at a time, ranging from bytes to a few terabytes	1. More costly to maintain in comparison with competitors	√
	Web Application	1. Can be used by any IOT device with a web browser 2. Much easier development process, testing only needs to be done on different browsers and screen sizes	1. Only accessible when connected to internet 2. Only accessible via a web browser (more overhead for frequent users)		Ionic	1. Runs the most consistently throughout all desired platforms 2. Development process is easier because of this consistency 3. Many built in UI features within the framework	Deeper customization to native devices is much harder to achieve	√	Microsoft Azure	1. One of the most economically viable options for storage space 2. Reliable and trustworthy enough to transfer data securely and efficiently on the scale of our project 3. Can scale amount of storage space for project sizes, from large multi billion dollar company programs to small group projects	1. While extremely reliable, can be somewhat limited in terms of usage	
	Progressive Web Application	1. Is accessible from any IOT device with a browser 2. Can be downloaded/stored as a mobile application on both iOS and Android devices. 3. Ability for the user to have an offline version of the application	1. Difficult development process since testing must be done on all 3 platforms (iOS app, Android app, PC web browsers) 2. Our team has the least experience with PWA development	√	React	1. Allows for more customization to native device 2. Is seemingly easy to use/pickup 3. Testing & Build process is made very easy with Expo	1. Code base doesn't as easily between platforms 2. Development process can be more difficult for a PWA		Google Cloud	1. Can be cheaper than AWS in specific instances, depending on what services are chosen 2. Allows for fast and easy accessibility from anywhere in the world Does not have a minimum size for objects being transferred, can select as much or as little data requested	1. Very simplistic in terms of storage and tools, does not provide much outside support other than the basics 2. Many instances it becomes more costly than AWS and may offer lower quality service	
					Flutter	1. Newer possibly "up and coming technology" for cross platform development 2. Offers great mobile application performance	1. Uses a coding language called Dart that our team has no experience with 2. Fewer online resources for troubleshooting/learning 3. Cannot yet build web applications		Kumulos	1. Helps to report crashes that occur when transfering data 2. Has many smaller features included in it's own SDK that can help to develop and fix problems	1. Mostly aimed towards standard mobile application development 2. Less known than the other options, so less community development and support	

Table 13: Pros and cons for each of our design decisions. Discussed are the possible options for platform development, UI frameworks, and data storage.