Ecolocation

Technological Feasibility

November 13, 2017

Client/Mentor:
Dr. Chris Doughty

Team Members:
Brenden Bernal
Chandler Hayes
Michael Hartzell
Anthony De La Torre

Table of Contents

1. Introduction

Animals play an indispensable role in providing for the overall health of our ecosystems. Of all the animals that play a role in this, megafauna (large mammals) play a disproportionately larger role due to their size, range of travel, and longer food passage times [1]. Due to the greater impact of megafauna, our project places its focus on these large mammals and seeks to provide information on the impact they have in a specified area. Our client and mentor Dr. Chris Doughty has been doing research in this area and we are working with him to make this information more accessible and understandable.

Although this information is accessible for existing megafauna, it is not readily available or formatted in a way that is easy to interpret. We are creating a unique mobile application that seeks to bring this information together and explain the data in ways that allow the user to understand the results. Furthermore, we will be providing information on megafauna going back thousands of years; mammals that are long extinct. This is a unique aspect of our application as range maps and nutrient distribution information on extinct megafauna is not publically available. We will be combining information on current and historic megafauna to provide unique data such as the effect that an extinct mammal could have on our ecosystems should it return.

In this paper, we will be exploring the significant technological challenges that will be encountered during this project and seek to find a solution. For our project, we will need to store and access large amounts of data and then use that data to run calculations and generate charts and graphs to represent the results. In order to accomplish these tasks we have chosen to develop an Android mobile application using Android Studio. For data storage we will be using **Amazon Web Services (AWS)** to host the data and Google Drive to store the photos we will be using. Finally we have decided to use MPAndroidChart for displaying the data. The following sections will address our chose solutions and how and why we decided on these specific technologies.

2. Technological Challenges

The creation of an application that will give users information about nutrient distribution contributed by megafauna in their local ecosystem leads to several technological challenges. The following are the technological challenges identified for the application:

- Easy to navigate design.
- Receive the user's location to give the user information on their local ecosystem.
- Given a location specified by the user, provide information on the ecosystem in that area
- Take large amounts of information about megafauna off of Wikipedia, like images and short descriptions, and put it in a database.

- Access to a database that will give it information on megafauna in various ecosystems.
    - The information includes:
        - A picture of the megafauna
        - A summary of the megafauna
        - The endangered level of the megafauna
        - The location of the megafauna
        - The weight of the megafauna
- The ability to calculate the nutrient distribution of megafauna in an ecosystem using the data given by the database.
- Display the data about an ecosystem to the user visually.
- Give the users an interactive way to manipulate variables in their ecosystem.

3. Technological Analysis

In the following sections, we will discuss the four main technological challenges. We will explore different solutions for each challenge by analyzing its features and comparing it with the needed features and functionally. Each section will explain its selected solution. Following this, a table with the alternatives is presented to compare them with the required features. Each section will conclude with its feasibility. The succeeding sections will cover mobile environments, storage and data hosting options, data visualization, and image storage and access.

3.1 Mobile Environments

Introduction

The mobile environment is the core piece of a mobile app and affects the available technologies that can be integrated with it. Ultimately, we selected **Android** as the mobile environment because it fulfills most of our needs. The first requirement is collaboration with our chosen graphing library, MP Android Chart. Additionally, the environment needs to have an adequate documentation and an active community. A decent documentation is beneficial for reducing the learning curve, effective use, following standards, and solving bugs more quickly. An active community will ensure that the libraries and other features will stay up to date as time progresses. A final component that would be beneficial is if the environment is able to collaborate with MATLAB. Our client uses it for computations and plotting graphs. We would like to leverage some of the MATLAB code for our application. In the following sections, three mobile environments will be explored: Android, Kivy, and Apache Cordova.

Options

Android

The **Android software development kit** (**SDK**) is one of the leading platforms for Android mobile development. It is usually paired with Android Studio which is the most prominent **integrated development environment** (**IDE**) for Android. The IDE has some valuable features such as debugging and instant feedback on layouts and **user interfaces** (**UI**). Android has a very active community. This means that provided libraries and frameworks for Android are more robust and the documentation will be detailed.

Pros:

- Strong and active community
- Android Studio improves the development process
- Native to Android phones will result in faster processing
- Can use our chosen graphing library: MP Android Chart
- Uses Java which is compatible with the other selected technologies
- Compatible with **AWS** (Amazon Web Services),our selected data storage

Cons:

- Incompatible with MATLAB
- Working with the interface can be tedious
- The IDE requires a lot of processing power
- Working with UI components can require more work

Kivy

Kivy is an open source framework that creates applications for multiple platforms such as mobile and web applications. The primary language that Kivy uses is Python which would allow collaboration with MATLAB through the use of pyzo [2]. Additionally, Python is an excellent resource for scientific use because it has packages aimed towards that domain [3]. Unlike Android, Kivy does not have a specific IDE. This means that development would be in a text editor and runs through a command line [4].

Pros:

- Compatible with the selected storage and data hosting of AWS
- Plotly could be used for graphing
- Compatible with MATLAB
- Many options for developing the UI
- Working with widgets is simplistic and is easy to learn

Cons:

- Installation is extremely difficult
- Building for Android APK is complex

        - Virtualbox has to be installed to support the Android environment on Windows
    - Documentation is not detailed
    - Is not native to Android, which will result in slower performance

Cordova

Cordova is another framework that provides cross-platform support for mobile and web apps. It uses CSS, HTML, and JavaScript to develop applications. With the use of JavaScript, Cordova can leverage the native devices **application programming interface** (**API**) to use native features. One of the highlights of Cordova is that plugins can be added to extend the code and use native device capabilities [2]. This can improve performance over an app that is not native to the OS. Although, it will be slower than an application that runs natively.

Pros:

- Builds applications for cross-platform use
- Installation and usability is simple
- Compatible with **AWS** (Amazon Web Services)
- Plenty of plugins for different elements
    - Such graphing or accessing the user's location
- Performs faster than entirely non-native mobile apps

Cons:

- Easy to produce a non-native experience
- Plugins are not guaranteed to be fully functional
    - Bugs would have to be manually fixed
- If a needed feature is not available, we would have to create the plugin
- Documentation can be poor and difficult to understand

Chosen Option:

0 = insufficient
5 = over qualified

Alternatives	Community & Documentation	Graphing	Usability	Compatible with MATLAB	Total
Java	**5**	**3**	**3**	**No**	**11**
Kivy	3	4	2	Yes	9
Apache Cordova	2	3	3	No	8

Table 1

- Android has an superior community and documentation
- Cordova plugins are not robust
- Android Studio has elements that make development easier
- Kivy is complicated to set up and to build projects

For the mobile environment, we chose to develop with the Android SDK. It had the highest rankings for nearly each category, which can be seen in *Table 1*. Its documentation and community excelled. These attributes are essential for producing a well rounded application. Due to having a more mature community, its libraries are going to be more reliable and the number of examples and resources are abundant. The communities and documentation for the other environments were weaker in comparison. For example, Cordova uses plugins to extend the capabilities of its applications. These plugins are vital in cross-platform development. However, a lot of the plugins can be unreliable since the community is still new and growing. This means that developers using Cordova will likely have to correct bugs in plugins or develop their own plugins. This is time consuming and requires extensive testing to ensure the plugins are fully functional.

The features of Android's **integrated development environment** (**IDE**), Android Studio, has various features that can make mobile development easier. For instance, the **graphical user interface** (**GUI**) for designing the **user interface**(**UI**) enables the developer to see the layout as they generate the UI. Another rare feature is that it includes a debugger. This can reduce time spent on debugging which is not available with Kivy and Cordova. Android Studio makes the development process smoother. On the contrary, Kivy has components that make this process arduous. The installation is complex and poorly documented. If the developer works in Windows, a virtual box must be installed. This causes building the project to be complex. This is undesirable because a project must be built every time the developer wants to test it. Ultimately, Android is the best option for our needs. It is compatible with graphing libraries and AWS.

Feasibility

We will create sample data of a reasonable size since the actual dataset is considerably large. With that dataset, we will demonstrate Android's capability to communicate with our databases: AWS and Google Drive. We will also experiment with different graphs on MP Android Chart. We will first access the data and run computations on the dataset. The results of the computations will produce graphs that reflect nutrient dispersal from all of the animals in the ecosystem.

3.2 Storage and Data Hosting

Introduction

Storing and accessing data is a large part of our mobile application, and the speed and reliability of our database will be a major factor in the efficiency and user experience of our product. To select the best option we need to make sure that it has the following features:

- Support for Android Studio
    - Ensure that Android Studio is supported because if we do not have the ability for easy integration and connection, then the rest of the features do not matter.
- Database migration
    - we have two databases, one for existing megafauna and one for the historic megafauna, and we need to be able to quickly and successfully migrate them to the hosting service.
- Backups
    - If data gets corrupted, or any other unforeseeable circumstance occurs we want to know that our data will be recoverable.
- Cost
    - we want an excellent service at an affordable price, but we do not want to sacrifice performance for the cheapest tool.
- Useful and thorough documentation
    - Good documentation leads to easier troubleshooting and better experience.
- Large user base.
    - Large user base is evidence of a good hosting service and with the large user base there will be additional help online aside from the official documentation that will prove useful.

Below we will examine three possible solutions to host our data: Amazon Web Services, Google Cloud SQL, and Microsoft Azure.

Options

Amazon Web Services (AWS)

The AWS tool that we are interested in using is **Relational Database Service (RDS)** which is a relational database management service that can work with MySQL for storing and accessing data.[5]

Pros:

- Android Studio support through **software development kit (SDK)**
- Tools that include database migration and backups
- Generous free tier
    - 5GB of storage
- Competitive pricing

- Thorough documentation and good tutorials
- Many users, a lot of good resources aside from AWS official documentation

Cons:

- Free tier lasts only one year
- Setting up seems more involved than Google or Azure
- Costs are complicated to understand

Google Cloud SQL (GCS)

Google Cloud SQL is a fully-managed relational database service that works with MySQL.[6]

Pros:

- Very good integration with Android Studio
    - Both Google Cloud SQL and Android Studio are google products so great support
- Database migration tool
- Cheapest base cost
- Good Documentation

Cons:

- Backups are an extra cost
- Free tier is not as competitive as AWS, does not provide functionality we need
- Smallest user base
    - Newest hosting service

Microsoft Azure

Azure Database for MySQL is a data hosting service designed for integration with applications.[7]

Pros:

- Support for Android Studio
- Quick and easy data migration
- Regular backups

Cons:

- Smallest free tier
    - 1GB
- Most expensive
- Documentation is not as easy to follow
- User base
    - Not as many users as AWS and slightly more than Google Cloud SQL

Chosen Option:

0 = insufficient
5 = over qualified

Hosting Service	Android Studio Support	Database Migration	Backups	Cost	Documentation	User Base	Total
AWS	**4**	**5**	**4**	**3**	**5**	**5**	**26**
Google Cloud SQL	5	5	1	3	5	4	23
Windows Azure	4	5	4	2	3	4	22

Table 2

For the final product, we are going to be using Amazon Web Services RDS for the following reasons:

- Full support with Android Studio
- Competitive price and usable free tier
- Overall ease of use
    - Ability to set it up and leave it alone
- Great documentation
- Good user base

AWS will be a great hosting service for our mobile application that meets all of our needs and does exactly what we need it to do. We will have to spend some time on the initial setup but once we have the data stored then we should be able to leave it alone and trust that it is safe and secure.

Feasibility

We have tested AWS with MySQL workbench and in a test application, and it worked well and was easy to connect to. The setup time was not very quick, but once we setup the database, it seems like the upkeep will be minimal and any changes that need to be made will be relatively straightforward. When accessing the data through the application it was swift and responsive with no lag time for connecting and responding to a query that we noticed. For further testing, we would like to move the databases to AWS and run some test with queries on a small subset and larger subsets to ensure that speed is still acceptable when accessing larger dataset. We also want to test it with multiple connections and queries being run simultaneously to see how it handles the increased traffic

3.3 Data Visualization

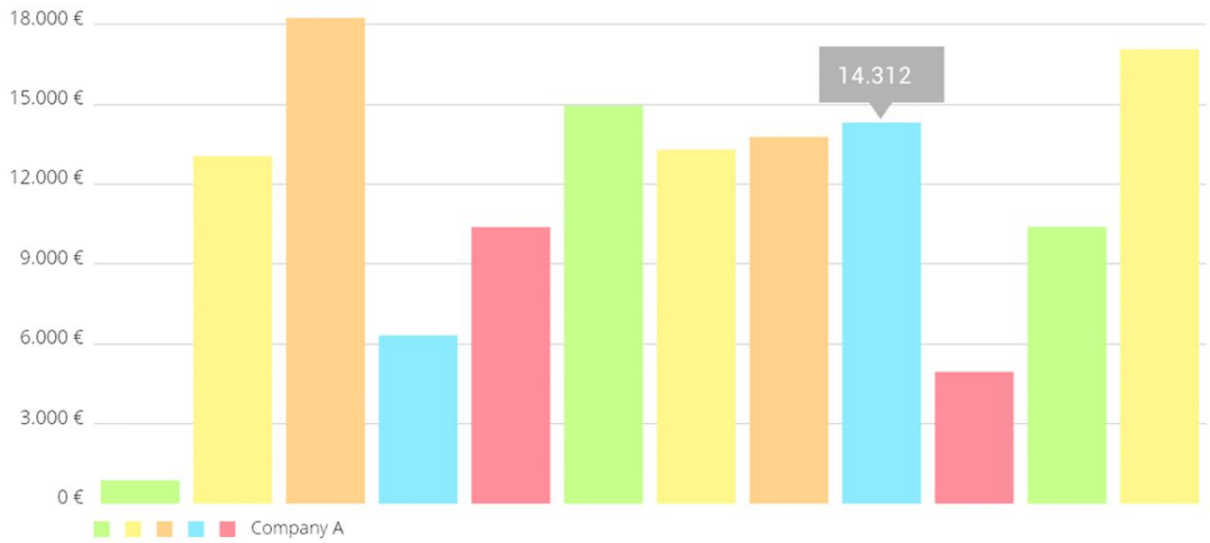
Introduction

In this section, we will look at our selected options for data visualization. In our application, we will be displaying several graphs that will summarize the data for the nutrient movement in the chosen locations ecosystem. We decided to look at MP Android Chart, Androidplot, and GraphView which are all Java libraries. In this section, we will give a summary of each alternative that will include our opinion on ease of use, functionality, documentation, and visual appeal. After each summary, there will be a list of pros and cons for the library. At the end of the section, there will be a table that will compare the three alternatives. Below we will analyze three graphing libraries: MPAndroidChart, Androidplot, and GraphView.

Options

MP Android Chart

Mp Android chart is an open source Java library that works for Android 2.2 and up. It supports several types of graphs such as line charts, bar charts, and candlestick charts just to name a few. This library also has some impressive features such as dragging and panning using touch gestures, the ability to combine different kinds of charts, a customizable popup views, the ability to animate the charts, and many other features. After viewing several tutorial videos for this library, we have determined that the library is easy to pick up and use. We found a lot of documentation for this library that will allow us to look up solutions if we run into problems while using the library. Here is an example of a bar chart created using the MP Android Chart library:



This graph came from MP Android Chart's GitHub page and the design for this graph is informative and easy to understand. [8]

Pros:

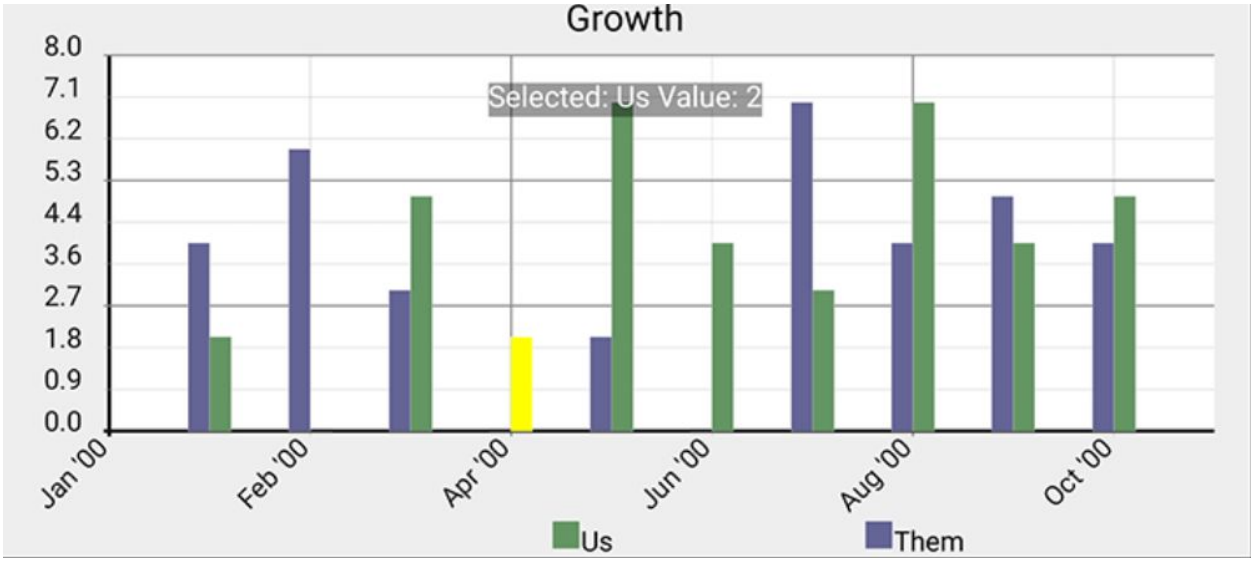
- Great style options
- Detailed documentation
- Works with Android Studio
- Many features to work with
- Many types of graphs to work with
- Regularly updated
- Easy to use

Cons:

- Can be slow compared to other libraries

AndroidPlot

AndroidPlot is an open source Java library that can run on Android 1.6 and onwards. It supports several charts like line charts, bar charts, pie charts, and candlestick charts to name a few. This library has many features like dynamic modeling support, the ability to mix types of charts, the ability to pan and zoom with the chart, and support for large datasets. AndroidPlot is used by over 1000 apps on Google Play according to appbrain.com [9]. The Androidplot library has detailed documentation making it easy to look up how to use the library. There is also an active community behind this library that makes many tutorial videos on how to use the library. The library is not as visually appealing as other options for this problem. The following image is an example of a bar graph using the AndroidPlot library to create it:



This graph was taken from AndroidPlot's GitHub page [10]. It looks dated and uninteresting but it is easy to get the needed information from it.

Pros:

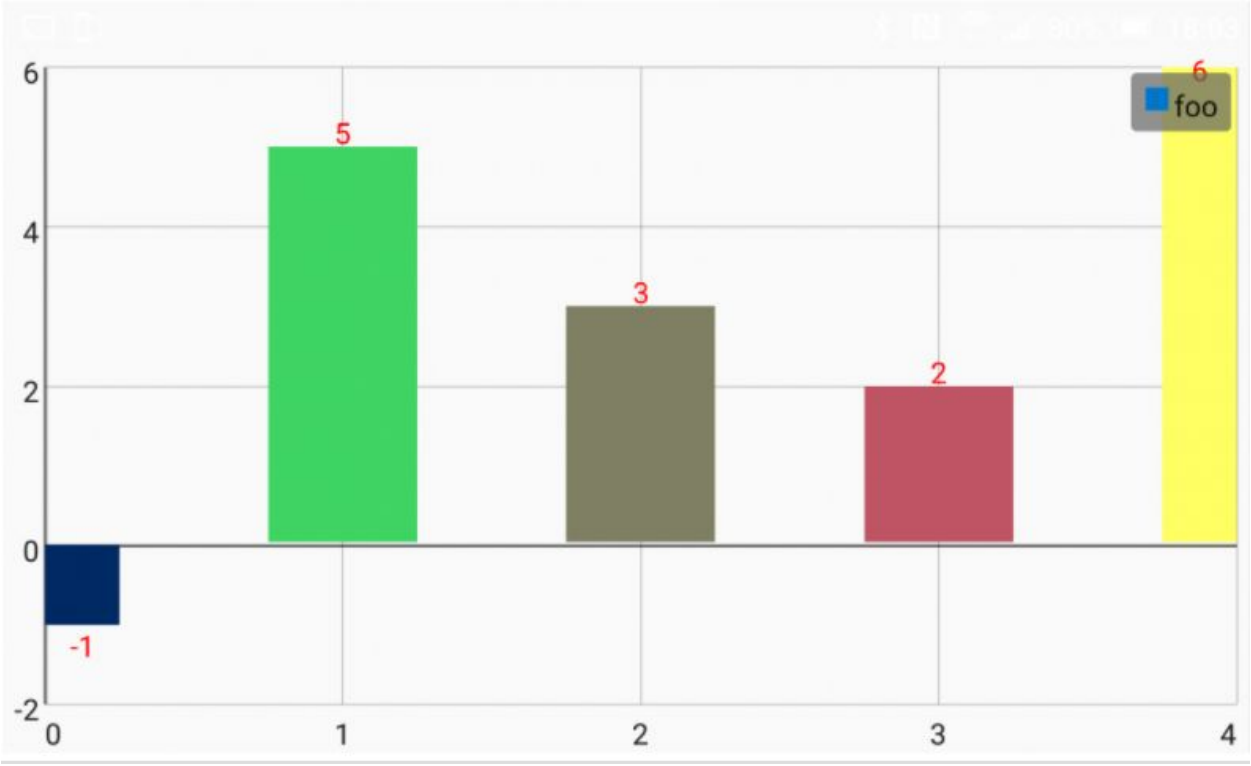
- Detailed documintation
- Easy to use
- Large array of charts
- Faster than other libraries
- Works with Android Studio

Cons:

- Charts don't come in many styles
- Not as many functionalities as other libraries

GraphView

GraphView is an open source Java library that is easy to understand and to integrate. This library has several charts that can be used like a line chart, bar chart, point chart, and it has the option of creating custom types of charts. This library comes with many features but not as many as other options. One feature this library has is a tap listener who can handle tap events on the charts which could allow the users to view more information on the data the chart is representing. Graph View has detailed documentation and tutorial videos on their website android-graphview.org that would make the library easy to learn and work with. This library is compatible with Android Studio and has XML integration. This is a Bar chart created using the GraphView library:



This graph was taken from the official GraphView webpage [11]. It is hard to understand because you can not add labels to the axis with this library for certain graphs.

Pros:

- Highly customisable
- A large array of functions
- Detailed documentation
- Easy to use
- Fast
- Works with Android Studio

Cons:

- Not as Visually appealing as other options
- Limited styling options
- Not scalable or scrollable

Chosen Option:

0 = insufficient
5 = over qualified

Alternatives	Ease of use	Functionality	Documentation	Visual appeal	Total
MPAndroidChart	**4**	**4**	**5**	**5**	**18**
AndroidPlot	4	3	4	3	14
GraphView	4	2	4	4	14

Table 3

- MP Android Chart has more functions than the other options.
- It has more documentation than the other option.
- It has better visual appeal than the other choices.

The library we decided to use was MP Android Chart. This library has detailed documentation and an abundance of tutorial videos that can be used to learn how to use this library and to use to find a solution to any problems we run into while using the library. The two other options we looked at, AndroidPlot and GraphView, also had substantial documentation and also had tutorial videos, but we did not find them to be as extensive or regularly updated as the documentation for MP Android Chart. The tutorial videos as well were not as detailed or numerous for the other two option as what we found for MP Android Chart.

The Second reason we chose this library is for the visual appeal. We found that MP Android Chart had the best visual appeal due to its many style options and fluid and bright designs. This library does suffer from speed for this look, but we believe that it is well worth it. The other two option's designs were not as appealing visually and did not have as many choices when it came to styles, but they are both faster. GraphView

design options seemed to basic and uninteresting to keep our user's attention, and the AndroidPlot library's design looks dated and dreary also has the least amount of customization when it comes to styles out of the three.

The last reason we chose to use MP Android Chart is its functionality. This library has functions like eight different chart types that will give us many options when it comes to displaying the data. It can drag and pan with touch gestures which will provide our users the ability to interact with the graph. The library has customizable popup-views which will allow us to offer our user's more information when they touch the graph. The library can combine chart types which will provide us with, even more, options for displaying data to the user. The graphs can have an animation which can be used to make the graphs look dynamic and more interesting to the users.

Feasibility

To test MPAndroidChart we made some simple graphs with data coming from a test SQL database. we add some animation to see how easy the feature was to work with which was fairly straight forward. The test was a success and we were able to implement a graph using the MP Android Chart on Android Studio.

3.4 Image Storage and Access

Introduction

One key feature for our app is being able to view information about the animals that live in a specified location. Such as their name, a brief description, and an image. For the initial phase of our app, we are restricting the database to about 5000 mammal species, and by storing the images for these species as jpeg files, they total a little over 1GB of data. As our app increases in scope, this number will increase as well, and our solution must handle this. Because the amount of data required vastly exceeds the average size for a mobile app, it must be stored externally. After this data is stored, the app must be able to access specific images as the user requests them, so access speed is a crucial factor. Other determining factors include Android Studio compatibility and ease of use. To make the best choice, we will explore the pros and cons of three different options: Google Drive, Amazon Simple Storage Service, and MediaWiki.

Options

Google Drive:

Google Drive is an online platform that allows for storage of up to 15 GB of data, including jpeg images, free. These files are stored in the cloud, but allow access at speeds similar to a local file system. You can use the **Drive Android Application Programming Interface (API)** to query and access this data. This API is native to Android Studio and is included when you download the **Google Play Software**

Development Kit (SDK) for Android Studio. To access the data your app must be registered with the Google API Console, which is complicated to set up; however, because this is a native app for Android Studio, there should be no problems after this initial phase.[12]

Pros:

- Adding the API to Android Studio is simple
- Built in functions for finding and accessing data
- Allows for access similar to a local file system
- Code to search drive is simple

Cons:

- Getting access to the drive from an application is complicated
- Only available for Android

Amazon Simple Storage Service (S3):

Amazon S3 is an online storage service that allows for storage and access of large quantities of data. The free tier puts restrictions on data size and is only available for a limited time. S3 includes an API that can be used for Android Studio and other Android development platforms. The installation of this API is simple, but apps require authentication similar to the Drive API. Since the API is not native to Android Studio, there might be some integration problems. Accessing the data through the API is straightforward, but connection speeds have been raised as an issue.[13]

Pros:

- API available for Android Studio
- Allows access to large amounts of data
- Easy to use

Cons:

- Free tier limited in time and data
- Issues have been raised about transfer speed
- Integration might cause problems

MediaWiki:

MediaWiki is an API that allows you to query Wikipedia pages for different pieces of information. There are several different options available, and Jwiki seemed like it suited our needs the best. It is a Java library that allows access to the MediaWiki API uniformly. Since it is querying a page directly, access to the information should be relatively fast. Getting the information from Wikipedia means that we do not have to store this data ourselves, eliminating any storage needs. Although there is some structure to the searches, you may have to parse through results to get to the desired information, which takes time. There is also no guarantee of the location of an image on a page. This could result in multiple searches to get info, and possibly not finding it. Since Jwiki is a Java library, integration with Android Studio should work, but there are no guarantees.[14]

Pros:

- Multiple versions of the API for different platforms
- No storage necessary for our app
- Access of the data should be fast

Cons:

- Parsing through search results to get desired info takes time
- The images might not always be in the same section of the page
- Integration with android studio is untested

Chosen Option:

0 = insufficient
5 = over qualified

Solution	Speed	Data restrictions	Usability	Android Studio Support	Total
Google Drive	**5**	**4**	**4**	**5**	**18**
S3	4	3	4	4	15
MediaWiki	4	5	3	3	15

Table 4

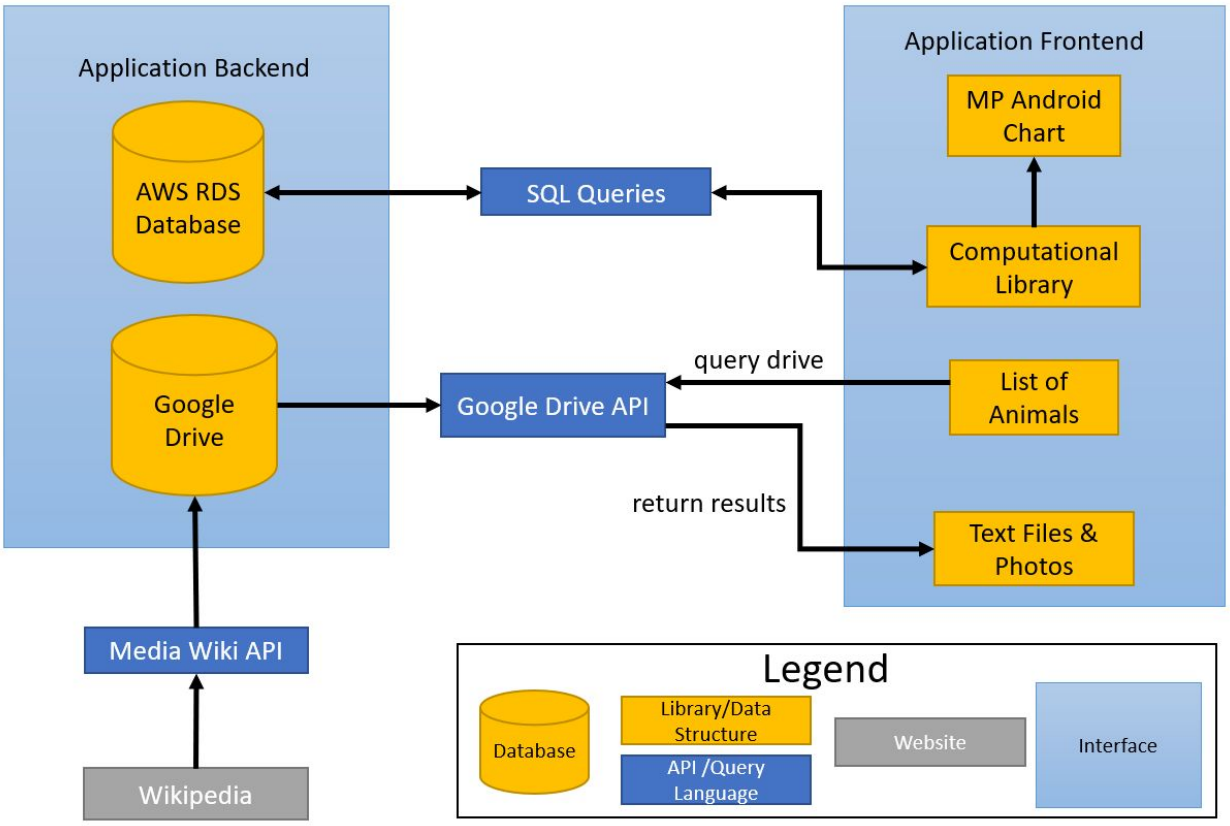
We decided to go with the Google Drive API. The most important reason for this decision is that the Drive API is native to Android Studio, the development platform that we decided to use. Android Studio is a platform that is known to have some integration issues and to have a solution that is native to this platform limits these issues. The second determining factor is the data restrictions imposed by the different platforms. Amazon S3 does offer free tiers of their data storage, but they only last a limited time and charge if data usage exceeds allocated amount. Google Drive allows for more storage than we will need for this app at no cost and with no time restrictions. This means that we will not have to worry about being charged for data usage. Another important reason for the choice is speed. Google Drive is a file storage system that We have been using for many years and access speeds have never been an issue, especially for small things like jpeg images and text documents. If these speeds hold for multiple users accessing the same drive, Google Drive should be our optimal solution.

Feasibility

To prove the feasibility of this choice, we did some research into the documentation that comes with this option. There are many different tutorials and guides available both through Google and third parties. We used some of these tutorials to follow the basic steps of installing the Google Play SDK and adding it to the compile dependencies. After this, you must add the authentication, which requires adding special keys and identifiers to your app. We were unable to achieve this step in the initial phase, but a little more time with the tutorials should get this part functioning. After your app has acquired authentication, all that is left is writing the actual code.

Although the MediaWiki API was not chosen as our solution, we might be able to use it in another part of our development. Because the short descriptions and photos were not given to us, we have to gather this information before we can store it anywhere. Because of its ability to access this info from Wikipedia, the MediaWiki API will greatly decrease the time to gather this information, but its lack of structure makes the risk of getting the wrong information too high for it to be our main solution. Using the MediaWiki API to gather the original information allows us to error check before storing it in Google Drive.

4. Technological Integration



With the selected technologies, we can assemble these pieces together. For the mobile environment we are using Android **Software Development Kit** (**SDK**). For the storage of species information we are using **Amazon Web Services** (**AWS**). To store the pictures and species description we are using Google Drive. To display the graphs we are using MP Android Chart. Java and the Android SDK are compatible with other three technologies and will be the interface between these components.The figure above demonstrates how these pieces will work together.

The first component is the *Application Cloud Base*. It consists of two different databases: AWS **Relational Database Service** (**RDS**) and Google Drive. The AWS RDS will store the information on existing and extinct species which are extracted from the **International Union for Conservation of Nature** (**IUCN**) and Historic Databases. The Google Drive will contain the images and short description for each species in the AWS database.

The second part of the system is the *Application Frontend*; this is the part of the app that the user directly interacts with. The app will generate a list of animals in the user's location. The list

will be used for computing nutrient dispersal so that it can be graphed with MP Android Chart. Furthermore, the user will be able to see the list of animals in their ecosystem. The list will contain an image and description for each animal on the list.

The integration between the *Application Cloud Database* and the *Application Frontend* will be connected by **Application Programming Interfaces** (**API**) and the **Structured Query Language** (**SQL**), The application will begin by querying AWS using SQL to produce the list of species and their corresponding information. After computations are applied, MP Android Chart will graph the nutrient dispersal and range maps. Additionally, the application will be able to present this list of the species so that each list item has a picture, a short description, and other information (like threat level). The image and description will be obtained from the Google Drive through the Google Drive API. The data on the Google Drive will be taken from Wikipedia by the use of MediaWiki API.

The integration between these technologies is relatively simple. On the *Application Frontend*, the four components are written in Java and will be compatible with Android. Communication between the *Application Frontend* and the *Application Cloud Database* are executed through the use of SQL and the Google Drive API. The Google Drive is able to get its data from Wikipedia by using MediaWiki API. Therefore, the use of APIs and the SQL language integrates the frontend and backend of the application making the integration of the technologies possible.

5. Conclusion

Chosen Options:

- Mobile Environment - Android Studio
- Data Storage and Hosting - Amazon Web Services
- Data Visualization - MP Android Chart
- Image Storage and Access - Google Drive

Animals play a role in many different services that are vital to ecosystems, such as nutrient and seed dispersal. Sadly, the awareness for the services that animals provide is lacking. To increase awareness for these services, we are going to create a mobile application that allows users to see the impact of animals on a given ecosystem in an easily understandable fashion.

To create our app we had to make many choices, the most important of which was deciding to use Android Studio as our development platform. Its built-in tools help development, and it can be integrated with our other software choices with ease. Another key decision was to host all database information on Amazon Web Services. This keeps all the information in one place and allows for quick access over the internet. In order to create and display all the required graphs

and charts, we decided to use MP Android Chart. This choice comes with an extensive library and allows us to display the information in a way that users will want to engage with. The final decision was using Google Drive to store the animal images. This choice gives us ample free storage with access speeds similar to a normal file system.

The final problem that we face is integrating these technologies together. Because most of our choices considered integration with Android Studio as a factor, the integration process should be without too many issues. We believe that we have made the correct technical decisions to create a great app that allows all the parts to be integrated smoothly.

6. References

1. The legacy of the Pleistocene megafauna extinctions on nutrient availability in Amazonia. Retrieved October 24, 2017. https://www.nature.com/ngeo/journal/v6/n9/abs/ngeo1895.html
2. Python vs Matlab. Retrieved October 22, 2017 from Pyzo · http://www.pyzo.org/python_vs_matlab.html
3. Pypi Pymatlab 0.2.3. Retrieved October 22, 2017 from Python: https://pypi.python.org/pypi/pymatlab
4. Kivy: Cross-Platform Python Framework. Retrieved October 22, 2017 from Kivy: https://kivy.org/#home
5. Amazon Relational Database Service (RDS). Retrieved October 24, 2017. https://aws.amazon.com/rds/
6. Google Cloud SQL. Retrieved October 24, 2017. https://cloud.google.com/sql/docs/
7. Microsoft Azure. Retrieved October 24, 2017. https://azure.microsoft.com/en-us
8. MP Android Chart. Retrieved October 22 2017 from GitHub: https://github.com/PhilJay/MPAndroidChart
9. Androidplot. Retrieved October 22, 2017 from AppBrain: http://www.appbrain.com/stats/libraries/details/androidplot/androidplot
10. Androidplot. Retrieved October 22, 2017 from GitHub: https://github.com/halfhp/androidplot
11. Showcase. Retrieved October 22, 2017 from android-graphview: http://www.android-graphview.org/showcase/
12. Getting Started. Retrieved October 23, 2017 https://developers.google.com/drive/android/get-started
13. Getting Started. Retrieved October 22, 2017 from Amazon: http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html
14. Jwiki Fastily Library. Retrieved October 23, 2017 from GitHub: https://github.com/fastily/jwiki