Technological Feasibility

10/11/2021

Team Biosphere

Project Sponsors: Jenna Keany, Christopher Doughty

Team Mentor: Melissa D. Rose

Team Members: Matthew Nemmer, Brandon Warman, Teng Ao, D'yanni Bigham

Table of Contents

1. Introduction

Overview

The loss of tropical forests is causing massive damage to the global environment. This is due to the vital role they play. By absorbing vast quantities of CO2, forests provide oxygen and help stabilize the Earth's climate. Furthermore, tropical forests help to maintain the world's water cycle through transpiration. The clouds they generate travel far and are responsible for rain all over the world! As for the flora and fauna that live there, "tropical forests contain over 30 million species of plants and animals. That's half of the Earth's wildlife and at least two-thirds of its plant species!" [1]. So, it is clear that these forests need to be preserved and protected. This is carried out by researchers, lawmakers, and climate activists, groups that are currently struggling to get accurate information on the subject. This is because the data they need is captured by satellites equipped with light detection and ranging (LIDAR) sensors, such as the International Space Station (ISS). Such data is difficult to acquire and visualize. The aforementioned groups often lack the technical skills required for this process.

Problem

It is crucial that data analysts and researchers are able to interpret LIDAR data about tropical forests. They use elevation and canopy height values for determining where specific species of plants and animals may be. Additionally, the results of the data allow policy makers and conservationists to bring about change and protect the forests. These groups are more interested in gaining an understanding of above-ground biomass. This helps them gauge how much carbon is stored within the forest, and therefore the forest's ecological importance and the environmental cost associated with its potential removal.

One way data is currently being interpreted is through Google Earth Engine (GEE). GEE is an online platform that allows public users to access and visualize geospatial data from numerous datasets, including the previously mentioned LIDAR data. While this platform is great, it is not readily accessible to users in the field. What local researchers need is an easy way to access environmental data and relevant maps in real-time while conducting their fieldwork. Such a tool is what the sponsors of this project are hoping will be created. They work under NAU's Megabiota lab, studying megafauna such as elephants living in the tropical forests of Africa. They're sponsoring this project in hopes that it will help them, their associates in Gabon, and the previously mentioned groups with their research.

Solution Vision

The ideal tool would be a mobile application capable of displaying data on a map as GEE does. This app's interactive heat map will present the data in an understandable way. The user can scroll and zoom the map, allowing them to view any region of interest they desire. Furthermore, they should be able to select the type of data they are looking at, be it canopy height, elevation, or above-ground biomass. They will be able to specify a range of values and areas with pertinent data will be highlighted. Finally, the user needs a way of specifying a region on the map that should be downloaded for offline use. This will greatly help researchers in the field when they cannot access the internet. In general, all the features stated should involve relatively fast load and response times.

Since the project is still in the early stages of development, the team has many many crucial decisions to make before diving into its implementation. This technological feasibility document starts with a discussion of the challenges associated with said decisions. It then focuses on the analysis of each type of technology, exploring the options and determining which will work best for the project. Finally, these ideas are brought together by stating how the chosen options integrate and create a working product.

2. Technological Challenges

The general idea is to have the mobile application receive its data from a server. The server will communicate directly with a remote storage system for data. In order for the server to be accessible to the mobile application, the server must be on a hosting service.

In order to accomplish this, there are a few requirements. The team will need to implement a native Android application. It will need to have a user friendly interface and an overlay that's optimal for the user's screen. The team will need to implement a backend server that will behave as an interface between a remote storage system and the mobile application. The team will need a remote storage system that will hold regional data such as canopy height, elevation, and above ground biomass. Lastly, the team will need a hosting service to store the backend components of the server. Listed below are the major technological challenges that will need to be resolved during the development phase of the project.

- **Mobile App Design:**
  - Choose a native Android framework to develop the application
  - Display the map overlay to the user in a friendly and informative manner
  - Be able to provide partial functionality in an offline mode
- **Data Acquisition and Storage:**
  - Keep data readily accessible
  - Develop team familiarity with data acquisition and storage of it
- **Web Server Design:**
  - Implement an API that will communicate with the Android application and a remote storage system
  - Receive requests from the Android application
  - Respond accordingly with the requested data
- **Web Hosting:**
  - Ensure components can be maintained at a low cost
  - Ensure enough storage space will hold the dataset
  - Making sure the hosting service can handle an acceptable amount of mobile traffic
  - Improve the security of held data through the use of firewalls

In the following technology analysis, the team will review potential technological solutions to the corresponding challenges in order to fulfill the clients' needs.

3. Technology Analysis

3.1 Mobile App Design

3.1.1 Issue Introduction

To create an interactive, user-friendly interface, this mobile application will require implementation of several features including a dynamic map that allows for user input, overlay capabilities for data provided to us by the clients, and toggles that allow for different sets of information to be displayed to the user. These are important factors when trying to decide on which mobile application framework will fit the client's needs for a mobile application to make access to complex data easier and create useful data visualization layers for analysis.

3.1.2 Desired Characteristics

→ Performance

Performance is the biggest factor when choosing a mobile application framework. When using the application and navigating around the map, the team doesn't want a delay that would decrease the user experience. The two types of frameworks that reach these requirements are native applications and cross-platform applications.

→ Feature access

There are a couple of features built into devices that are required for the mobile application. The clients want the ability to access a device's GPS data, if available, to center the map on a user's current location. There is potential for a user to require offline access, which requires access to a device's local storage for relevant data to be downloaded.

→ Codebase

Deciding on a framework depends on the compatibility with the mobile application and the complexity of learning the language. The application requires a framework that allows for the creation of map views. In addition, it will require the ability to interact with the chosen backend framework.

3.1.3 Candidates

→ Native application

Considering the top priority as performance, a native built application is a clear option. Since a native application is written specifically for a device's operating system, it interacts directly with the device to increase performance and efficiency. Natively built applications also have access to a device's full features including GPS, storage, and device specific user

interface. When updates to a device's operating system occur, the changes are accessible immediately for use.

→ React Native

A company made by Facebook, React Native, is a cross-platform framework that allows for a codebase to be reused for iOS and Android applications. Notable companies that use React Native for their mobile applications are Instagram, Walmart, and Bloomberg [2]. React Native is written in JavaScript but is partially compiled into a device's native code. Because of this, the framework has increased performance and allows for full access to features by installing extra modules. Updates to a device's operating system aren't instantly added to the framework.

→ Flutter

Flutter is a company created by Google, the language for the framework is Dart which is like C#. There are large companies that use Flutter, notably Alibaba and Google [2]. Flutters code partially compiles into a device's language giving it increases to performance and full feature access. There is a growing community behind Flutter and an increasing amount of community backed modules. It tends to have higher performance and compatibility compared to React Native [3].

3.1.4 Analysis

Performance

Considering performance as a top priority for the mobile application many factors need to be considered for the framework. An important area will be rendering images and a map to display on the devices screen. In terms of battery usage, a native Android application is around 70% more efficient in battery usage while outperforming cross-platform applications. For iOS applications the difference in efficiency is around 80% [3]. A native application significantly outperforms cross-platform.

Feature access

Native applications have the best integration with a device and allow for offline use of features. Cross-platform applications require an API connection that communicates with the device over a network connection. There are modules, such as React Native Mapbox that supports offline functionality but is no longer actively maintained. Due to this, a native application remains a great choice.

Codebase

A consideration when deciding on a framework is the complexity of the codebase. Being able to reuse code for an Android and iOS application is a perk for cross-platform frameworks. In testing, it turned out to require a significant amount of code specific to each operating system. With a native application it must be written in Swift for iOS and Java for Android. iOS development is only possible to perform on a Mac computer, which limits the team's ability to create one. This is the same if a mobile application is cross-platform or native. To meet all of the requirements, the team will be using a native application for Android, and for the clients' stretch goal a native iOS application.

3.1.5 Chosen Approach

Native applications meet all the requirements the team needs for a mobile application. Having significant performance increases over a cross-platform application and offline functionality of maps and GPS data. React Native and Flutter share portions of the codebase between devices which native applications cannot.

Criteria	Performance	Feature access	Codebase	Weighted Total
Weights	0.5	0.3	0.2	1
Native	5	5	2	4.4
React Native	3	3	4	3.2
Flutter	4	3	5	3.9

3.1.6 Proving Feasibility

The team will be starting with a native Android application to prove feasibility. This is due to many factors but most importantly is that the team is able to test the product during production. Once an Android application has been successfully developed, a stretch goal for the team will be to develop an iOS app. Due to the improved performance, increased response time, and offline functionality makes a native application the clear choice when deciding a framework to use. The only downside is that the codebase for iOS and Android would be completely different, requiring more time to implement both.

3.2 Data Acquisition and Storage

3.2.1 Issue Introduction

The retrieval of data by the mobile application is vital to its functionality. It enables the app to effectively convey this data to the user via a map. The simplest way to accomplish this goal is by using map tiles, which are commonly formatted as raster images such as GeoTIFFs (.tif), Bitmaps (.bmp), and JPEGs (.jpg) [4]. So, the team requires a place where these images can be stored and retrieved from, such as a database or a file system. Once the mobile device has acquired the images it needs, they will be kept in main memory for actively displaying the map, and in local storage when needed for offline use.

Additionally, the issue of where this data will be initially acquired from needs to be addressed. Large quantities of satellite imagery are currently stored in Google Earth Engine (GEE), a platform that provides global analysis capabilities using these datasets. The images needed for this project will be exported from GEE as GeoTIFFs to a team member's Google Drive. From there, the files will be transferred to the chosen storage location.

3.2.2 Desired Characteristics

It should be noted that all the storage mediums discussed are free to use.

→ Provided Toolset

Each method of data storage will come with its own set of tools that can be used. The most useful features will assist with organizing and maintaining the project's data. These are important as the map tiles need to be stored in a structured way, be kept secure, and maintain their integrity.

→ Ease of Data Retrieval

How difficult it is to retrieve data will directly impact the mobile app's performance and the time spent implementing the app and server. Having simpler queries and requests to write will yield more time for the team to work on more important tasks. Furthermore, the requested images need to be retrieved quickly to provide fast response times for the interactive map.

→ Implemented Language

Each storage medium being considered requires the use of a specific language. These languages have their respective syntaxes and nuances. The ones that are easier to use will typically require less lines of code while still being human-readable. The ideal language will assist the team with reducing the number of errors and mistakes made. Another factor to consider is the team's previous experiences with these languages.

3.2.3 Storage Candidates

→ MySQL

MySQL is a relational database, meaning it is ideal for storing structured data in tables and is capable of utilizing the relationships between them. It's structured query language (SQL) and syntax are considered simple and fairly human-readable when compared to other forms of SQL. It includes many Spatial Analysis Functions that are useful for geographic information systems (GIS) like this project [5].

→ MongoDB

MongoDB is a non-relational database, also called a NoSQL database. This type of database management system is considered document-oriented. It is better suited for non-structured data and can be managed using numerous different programming languages. When compared to relational databases, it is simpler to query and does not follow the same tabular schema. It also features geospatial queries for GIS applications [6].

→ PostgreSQL

PostgreSQL is another relational database. However, it is further specified as being object-relational, meaning it can store program objects as well as relational data. It is able to utilize the extension PostGIS, which has useful features and functions for handling GIS data [7].

→ File System

A file system is simpler than a database in that the images can be accessed directly. It will be part of the project's server and rely on its language for its minimal set of operations. Unlike databases, file systems are managed by the operating system it runs on and thus requires minimal work to maintain it.

3.2.4 Analysis

→ Provided Toolset

Databases provide tools that file systems do not. They are more capable of managing and organizing data. However, the GIS tools mentioned for the three candidate databases only work primarily with GeoJSON files, which are not the ideal format for mobile application map tiles. So while the databases still provide more functionality, they do not necessarily provide tools specifically useful to the project.

→ Ease of Data Retrieval

For databases, data retrieval is accomplished through queries. Due to their syntax and structure, these queries are typically more complex for SQL databases such as MySQL and PostgreSQL than they are for NoSQL databases like MongoDB. File systems however do not

require any queries. They can access files directly and are better suited for this purpose than databases, which are ideal for raw data. So, a file system can access images faster while requiring less work than a database.

→ Implemented Language

As previously mentioned, file systems take on the language of the server. Despite that all team members have worked with databases to some degree, it is not required that the team learns a new language for it. Understanding the language of the server will be sufficient. Furthermore, file systems rely on very few instructions for data retrieval, making the process simpler overall.

3.2.5 Chosen Approach

Through this analysis, the team determined that a file system would likely be a better storage medium than a database. Despite not including a variety of tools to use, a file system is able to access images faster and is easier to implement than a database. With it being an integral part of the server, the language will

The options are rated below on a scale from 1 to 5, with higher numbers representing a better score for the given criteria.

Criteria	Toolset	Ease of Data Retrieval	Language	Weighted Total
Weights	0.3	0.5	0.2	1
MySQL	3	3	4	3.2
MongoDB	3	4	4	3.7
PostgreSQL	4	3	3	3.3
File System	2	5	5	4.1

3.2.6 Proving Feasibility

The storage medium used for the project will be the file system present on the chosen server. For the Technology Demonstration at the end of the semester, the team will create a prototype mobile application. This mobile app will be able to request data from the server, which will access it's file system and return the data. These map tiles will be stored on the local device, either temporarily as cached data for actively displaying the map, or in storage for offline-use

3.3 Web Server Design

3.3.1 Issue Introduction

The mobile application is going to need to have access to a remote storage system to retrieve the necessary data. It is not ideal for the app to ask the user to store the data locally on their device, this is not ideal for two reasons. The first reason is physical storage on the device will not be large enough. The second reason is there will be extraneous data if stored locally. The best solution to this will be to have a web server communicate to a remote storage system for data. The server will act as a "middleman" between the app and the remote storage system. In order for this to work, the team is going to need to create a REST Application Programming Interface (API). The app will send API requests to the server on what data is needed, the server will then respond with the requested data.

3.3.2 Desired Characteristics

→ Language

The language that the candidate is in plays a part in how the rate of development is going to be. If the language is known by the team, then the team can start implementing the application as early as possible. However, if the language is not known by the team, then they will have to spend time learning it. The ideal candidate has to be in a language that the team is most familiar with.

→ Documentation

Documentation needs to be written well for the following candidates, as this will be the main source of help or guidance. The documentation must include a "getting started" section or an equivalent to create a foundation for the server. It also needs to include a reference that will showcase how all the functions work and examples of them. The ideal candidate needs to have comprehensive documentation

→ Previous Experience

Previous Experience with the server framework is also a factor for the development rate. If the team or even one teammate has prior experience with this. They might be able to give insight on how implementation will be or any pitfalls that might occur. The ideal candidate should be known by at least one person within the team.

3.3.3 Candidates

→ Express.js

Express. js is written in JavaScript. Most of the content viewed on the internet uses Javascript in some way. The purpose of Express.js is to bridge the gap between client and server side programming. Developers with knowledge of the client side can now write the

server side of their applications since it uses the same language for both hence the popularity of Express.js. It is primarily used for building web servers that serve content to the client whether it is web or mobile. A few big companies such as PayPal, Linkedin, and Capital One use Express.js for their backend servers.

→ Django

Django is written in Python and created by the Django Software Foundation (DSF). Django works with an "out of the box" philosophy so developers can spend more time creating a large scale application than worry about all of the intricacies of backend development. It is an excellent framework to use for creating full scale applications quickly. Some popular sites that use Django are Instagram, Mozilla, and Pinterest.

→ Spring Boot

Spring Boot is written in Java and many enterprises seem to favor Java due to its ability to scale and maintain projects. Spring Boot's philosophy is "add whatever module you need" ; this is a great approach since the backend might need specific modules. This was also considered since the team wants to develop a native Android application. Many large companies use Spring Boot such as Walmart and Udemy.

3.3.4 Analysis

→ Language

The method taken for this was everyone discussed their experience with the language. The team was experienced with all the languages these frameworks are written in. However, Java was not as favorable compared to Python and JavaScript. The reason being that the team hasn't worked with Java in a while. Which will take time to relearn.

→ Documentation

The method taken to assess this was to thoroughly go through the official documentation. Spring Boot's documentation contained a "getting started" which was great, but the downside of it was there was unfamiliar syntax and terminology being used to explain the code. The documentation does contain guides for how to do certain actions within the framework. However, the organization of the information is convoluted. In order to find specific information, one must click link after link to get to specific pages.

Django's documentation contained a comprehensive "getting started" section. The "getting started" guide starts from the basics of the framework and goes up to a functioning server. Also the documentation page is well organized as it has links to anything related to the framework. Unlike Spring Boot, it takes only one link to get to the desired information. It

also links to a reference page that contains every method in the framework. Its' reference gives examples for all of the methods it uses or how concepts work with one another.

Express.js's documentation contains an easy to follow "getting started" section. In this section it covers installation to getting a simple server running. The documentation also covers specific sections such as routing, middleware, and error handling that the team will utilize throughout development. Also, it is minimal in links, so it's rather simple to get the desired information. Because of the simplistic but informative documentation, Express.js was found to be the best.

→ Previous Experience

The method taken for this was the same to evaluate language experience. The majority of the team has stated that they've never worked with either Spring Boot or Django . However, the whole team has stated that they have previous experience with Express.js. It's because the team has had a class in the past where Express.js was used.

3.3.5 Chosen Approach

The options are rated below on a scale from 1 to 5, with higher numbers representing a better score for the given criteria

Criteria	Language	Documentation	Prev Experience	Weighted Total
Weights	0.3	0.5	0.2	1
Express.js	5	5	4	4.8
Spring Boot	4	3	2	3.1
Django	5	5	2	4.4

Express.js is the ideal solution for the backend server. It uses the most popular language on the internet: JavaScript. The team is most familiar with the language that Express.js is written in. Documentation was weighted heavily the most because it is going to be the first resource the team utilizes when they need help. Its documentation proves to be the most readable, but at the same time being simplistic enough to comprehend how to use its tools. Outside of the official documentation, there are plenty of resources available. It also helps that the whole team has previous experience with this framework. This is the most promising candidate for the backend server.

3.3.6 Proving Feasibility

To prove feasibility, the team is going to use Express.js in the Technology Demonstration at the end of the semester. The plan is to show that this technology does in fact work with the Android application. To demonstrate this, the application will make simple API calls to the server. The server will then print a message to the app that its request has been received. Further in development, for a demo the server will send a small piece of real data. This will show that the team is able to successfully send data to the app.

3.4 Web Hosting

3.4.1 Issue Introduction

The Biomapper applications will contain big data of Africa and they all be stored in the same hosting, so the performance of the web hosting will be one of the most important parts in the circumstances because the data size of the map is really huge. The meaning for the web hosting is that it is the backend of the Biomapper and the Web hosting is the middle portal and it will store the map's big data from satellites and it can be pulled by users from web hosting. A good web hosting service can make the mobile application respond quickly and the users will also have a great experience of the mobile application. Web hosting also needs to be well structured and have enough security to make the data safe. All big map data and user information will be stored in the web hosting. Moreover, In the website or in different web hosting companies have different plans and different quality of the web hosting, and this part of the Technological Feasibility is to analyze the advantages of each hosting provider and each different kind of web hosting and choose the best web hosting.

3.4.1 Desired Characteristics

The desired characteristics of the Biomapper ideal web hosting will be stable security, enough available storage capacity, lower cost, and good performance.

And web hosting can be ranked by 4 factors: available storage, monthly traffic, data security, and Payment.

→ Storage

The project requires web hosting that can store about 500GB of data of a specific square of the map at the beginning, and a web host must be selected that can hold that amount of data.

→ Traffic

It is estimated how much data will be pulled from the web hosting and it also is charged by the amount of data that was pulled by the user.

→ Security

To distinguish between well-secured web hosting and poor secured web hosting, the firewall and protocol can be compared. The more advanced it is, the safer it will be.

→ Lower Cost

In the same amount of storage, same monthly traffic, and same security it has, the payment is the only way to help the team to pick up the web hosting. The team would like to pick up the cheaper one to reduce the cost for the clients.

3.4.3 Candidates

→ Google Cloud Platform

Google Cloud Platform is a powerful and stable platform. It has several advantages and different elements such as Google Compute Engine, Google Cloud App Engine, Google Cloud Storage, Google Cloud Endpoints. Etc. What the team needs for the Biomapper will contain 2 parts, the Google Cloud Compute and the Google Storage. These two parts can satisfy the application web hosting requirements and it also has a powerful firewall to protect the map data [8].

→ Amazon Web Services

Amazon Web Services is a powerful cloud service that provides services in different modularity blocks, and the different blocks can be used to organize the highly scalable level of applications in web hosting. And it has several parts of services like the AWS Compute, AWS-Storage, AWS-Database, AWS-Management Tools, and AWS-Migration [9]. In addition, The Computer Science department of NAU has a contract with Amazon Web Services. It makes it possible for the team to customize the web hosting through Computer Science and it is entirely free [10].

→ Bluehost

Bluehost is one of the most famous web hosting providers for beginners or official companies, it offers the cheapest plan for users and also has powerful web hosting for big companies. It also supports features and easy-to-navigate cPanel, and it will ensure the Biomapper learning curve is as small as possible and get the Biomapper application working as soon as possible [11].

3.4.4 Analysis

The Biomapper team created the standard to measure the 3 Candidates' dedicated Server. And it is easier to see which one is better.

The Standard:
Storage: About 500GB
Monthly Traffic: About 5TB
Data Security: Well structured firewall

→ Google Cloud Platform
Available storage: 500GB
Monthly Traffic: 5TB
Data Security: Well Structured Firewall
Payment: 627.15$ per month [12]

→ BlueHost
Available storage: 1TB
Monthly Traffic: 5TB
Data Security: Well Structured Firewall
Payment: 79.99$ per month [13]

→ Amazon Web Services
Available storage: Customize
Monthly Traffic: Customize
Data Security: Well Structured Firewall
Payment: Free

Based on the above information, it is clear that Google Cloud Platform has the highest price and that it is focused on multifunction web hosting. The more modularities that the team uses, the price for a single one will be cheaper. The Blue Host web hosting offers a good performance of the Web hosting which has 1TB storage, 5TB monthly traffic, and Free IPs, Domain for free.
But the Amazon Web Services option is a unique one. The team can customize the AWS web hosting through the "NAU AWS Web Hosting Contract." For the prototypes and software design process, the basic 8GB storage would be sufficient for testing. After the Biomapper application has been designed and pulls the big map data from web hosting, the team could upgrade the storage for officially working, and the web hosting will be free and it can reduce the client's cost.

3.4.5 Chosen Approach

The options are rated below(Figure 3.4.5) on a scale from 1 to 5, with higher numbers representing a better score for the given criteria.

WEB HOSTING						
	Criteria	**Available storage**	**Monthly Traffic**	**Data Security**	**Payment**	**Weighted Total**
	Weights	0.3	0.2	0.2	0.3	1
	Google Cloud Platform	4	4	5	1	3.3
	Blue Host	5	4	4	4	4.3
	Amazon Web Services	3	5	5	5	4.4

Figure 3.4.5

The WEB HOSTING decision matrix (Figure 3.4.5) shows that the team's best choice is the AWS web hosting and it also has the highest score. It is free and the team can customize the storage and performance to what is required.

3.4.6 Proving Feasibility

Choosing a web hosting company is not a technological problem for the team. But how to be satisfied with all the Team requirements and use the lowest cost is what is discussed in this section. To consider the lowest cost, enough storage space to hold the Biomapper dataset, good performance to hold the huge amount of mobile traffic, and secured data protection. The AWS is the team's final decision for web hosting selection, it is free and can satisfy the Biomapper all requirements.

Furthermore, the team plans to set up the AWS web hosting through the Computer Science Department at the end of the semester and test the Express.js server and its file system to prove the Biomapper technological feasibility.

4. Technology Integration

The Biomapper application will initially be developed as a native Android app to test all the functionalities. The backend software framework will be an Express.js server that is stored on the chosen web host, Amazon Web Services. Additionally, the file system present on the server will be used for data storage.

By having this technological component integrated into the team project all users will be able to use this application, and this will increase the total size of users that also has stable communication traffic throughout the AWS web hosting. For the Mobile application development part, the team will use the native Android to do the framework and test the functionality such as pulling data. And for the end semester's Demo will test the message sending functionality and receiving functionality.

The team will be starting with a native Android application to prove feasibility. Once an Android application has been successfully developed, a stretch goal for the team is to develop an iOS app. At the end of semester, the team will prove the technological feasibility by having a mobile app retrieve a small piece of real data from the file system, proving that it can be done successfully. This app prototype will make queries to the needed images via their respective URLs. Finally, the team will show that data pulled from the file system can be stored within the mobile device's local storage for offline use.

Below (Figure 4.1) is a system diagram that illustrates the plan for having all the chosen technological components interact with one another.

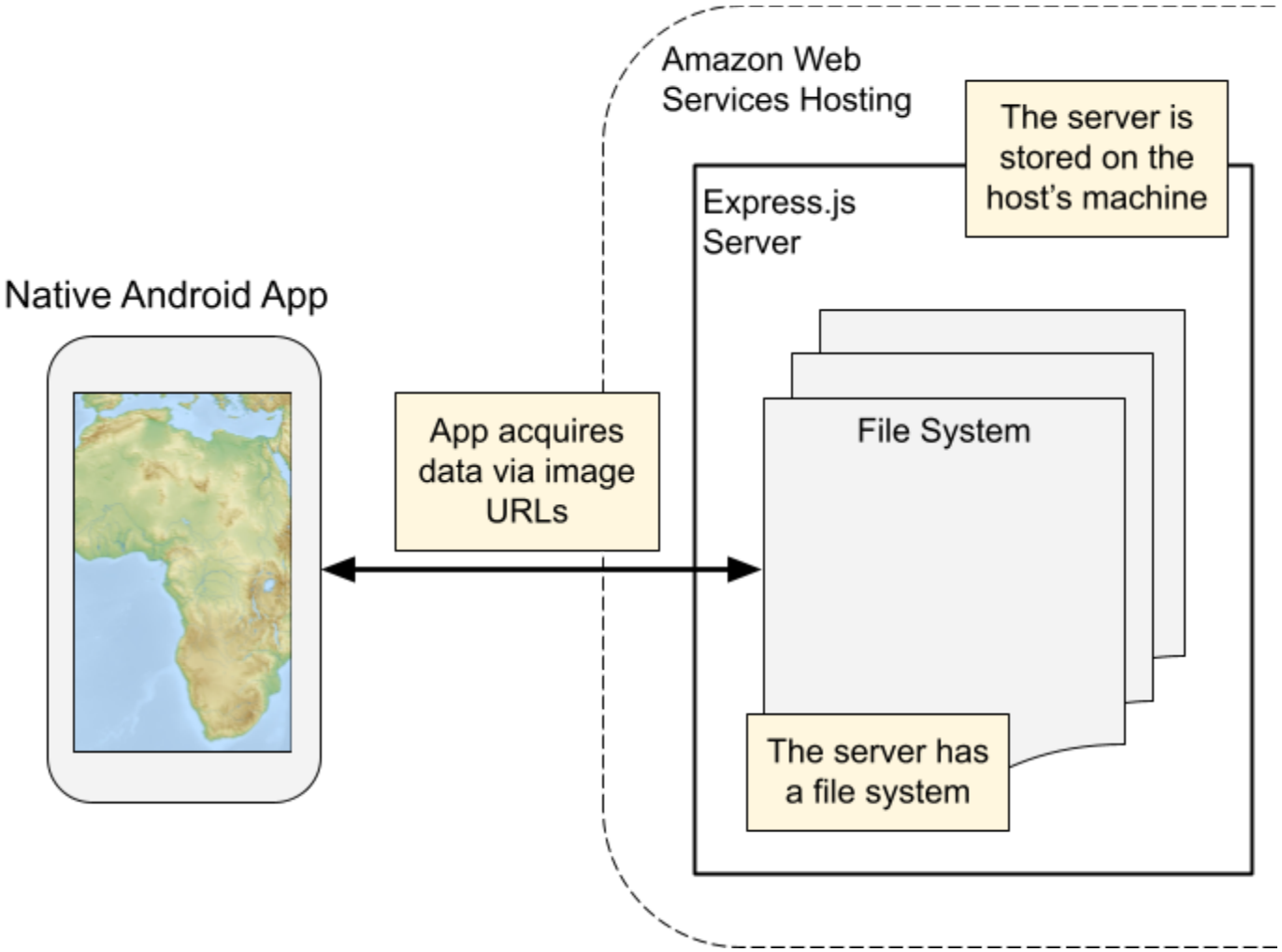
Amazon Web
Services Hosting
The server is
stored on the
host's machine
Express.js
Server
Native Android App
App acquires
data via image
URLs
File System
The server has
a file system

Figure 4.1

5. Conclusion

A critical role in the fight against climate change, geospatial data allows scientists to analyze data and see how a particular area is fairing. These data sets can include canopy heights, carbon storage, wildlife habitats, and other beneficial data. The problem with this data is the complexity of accessing, understanding, and being able to visualize the data. This is where the team looks to help the clients and their partners in Gabon. Scientists need access to these valuable resources while in the field and resource managers need an easily accessible version of it. Through the creation of a mobile application that will display preprocessed data visualization to a user's mobile device. Configurations for different overlays such as canopy height and biomass that can be filtered to a desired height. More specific information such as longitude, latitude, and canopy height/biomass in the area can be displayed to a user if they tap on the map. The client's current method for accessing the geospatial data is through Google Earth Engine, which is a web application that requires an internet connection to use. This can be improved on through the mobile application that will be able to download specified regions for offline use and the other features listed above. The clients will be giving us access to the GEE dataset that will be uploaded to a server for storage. Using Express.js the mobile application will be able to communicate to the server for data retrieval and storage. The team's decisions on frameworks were decided through testing and research of different frameworks factoring in performance, compatibility, and other factors.

6. References

[1] “Why are rainforests important?,” *Rainforest Concern*. [Online]. Available: https://www.rainforestconcern.org/forest-facts/why-are-rainforests-important.

[2] T. Bhatt, “Flutter vs React Native: An In-depth Comparison Between the two Frameworks,” *May 25, 2021*. https://www.intelivita.com/blog/flutter-vs-react-native/.

[3] I. Demedyuk, N. Tsybulskyi, “Flutter vs React Native vs Native: Deep Performance Comparison” June *29, 2020.* https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison.

[4] “Raster vs. Vector Graphics and graphic file formats,” *SITE IMPULSE Blog*, 12-Jan-2021. https://www.siteimpulse.com/blogen/raster-vs-vector-graphic-file-formats/.

[5] “MySQL 8.0 Reference Manual : 12.17 spatial analysis functions,” *MySQL*. https://dev.mysql.com/doc/refman/8.0/en/spatial-analysis-functions.html.

[6] “Geospatial queries” *Geospatial Queries - MongoDB Manual*. https://docs.mongodb.com/manual/geospatial-queries/.

[7] P.G.I.S. Developers, “Postgis” *Spatial and Geographic Objects for PostgreSQL*. https://postgis.net/.

[8] “Cloud Computing, Hosting Services, and APIs,” Google.com. https://cloud.google.com/gcp?utm_source=google&utm_medium=cpc&utm_campaign=na-US-all-en-dr-bkws-all-all-trial-e-dr-1009892&utm_content=text-ad-none-any-DEV_c-CRE_509008856042-ADGP_Desk%20%7C%20BKWS%20-%20EXA%20%7C%20Txt%20~%20Application%20Modernization%20~%20Application%20Modernization%20~%20Web%20Hosting_Pricing-KWID_43700061552833239-aud-388092988201%3Akwd-158433199235&utm_term=KW_google%20hosting%20pricing-ST_google%20hosting%20pricing&gclid=CjwKCAjwiY6MBhBqEiwARFSCPnFehJp4bUOGzFmKUST1QcZVANY6ul8i4rtvInuuKvUDXggTjxqX2xoCT-QQAvD_BwE&gclsrc=aw.ds.

[9] S. Nijim, “Learning amazon lightsail” *Amazon*. https://aws.amazon.com/lightsail/pricing/?nc1=h_ls.

[10] “AWS Hosting for Capstone” Nau.edu. https://www.ceias.nau.edu/~edo/Classes/CS_Capstone/Docs/AWS-hosting.html.

[11] “Dedicated server web hosting - best Linux servers - Bluehost” Bluehost.com. https://www.bluehost.com/hosting/dedicated.

[12] “Google Cloud Platform pricing calculator” Google.com. https://cloud.google.com/products/calculator/.

[13]“Dedicated server web hosting - best Linux servers - Bluehost” Bluehost.com. https://www.bluehost.com/hosting/dedicated.