Tech Feasibility

October 25th, 2024

Data Integrity and Abuse Prevention for Environmental Preservation

Team Members: Aidan Trujillo, Fady Zaky and Kyle Bambling

Mentor: Scott Larocca

Sponsor: Dr. Camille Gaillard, Dr. Duan Biggs, Dr. Jenna Keany, and Dr. Chris Doughty

Table of Contents

1. Introduction

In 2022, 180 of the world's governments came together to decide that they will, by 2030, conserve 30% of the earth's land and sea. This came in an effort to promote biodiversity across the globe. Since this is such a large portion of the world, there becomes a problem with equity for some communities that live in areas where conservation is taking place especially with endangered species. The people that live here can find themselves being displaced from the land that they may have lived for their entire lives. As important as biodiversity and conservation is for the world, the consequences are life changing for some. With that being said, it makes it much harder to link conservation efforts to communities when they are being directly affected.

Our clients, Dr. Camille Gaillard, Dr. Jenna Keany, Dr. Duan Biggs, and Dr. Christopher Doughty, have noticed the challenges that these communities face. In an effort to give back, that is where our project comes in. Due to a concept known as citizen science it is now a possibility for everyday people to contribute to research and data collection. Our clients want to create an application that is going to allow anyone from rural communities to contribute to citizen science so long as they have a mobile phone.

For our application, people in these communities will be able to contribute to validating data from the NASA GEDI satellite. The GEDI satellite collects data from around the globe and is able to determine ecosystem biomass and carbon content. This data of course needs to be verified and that is where citizen scientists come in. By using our application, users will be able to upload photos of the biodiversity in their area so we know what is actually there along with collecting measurements to assess the accuracy of GEDI satellite data. The end goal is being able to monetize this for our users so that their contributions made to citizen science are rewarded. In turn this should be able to funnel needed funds into rural communities affected by this while also motivating more affected communities to contribute.

An application developed to be monetized for users in remote areas comes with many challenges. The following sections will present some of the biggest challenges that we have identified as well as the solutions and technologies to help us overcome these.

2. Technological Challenges

A big challenge that we need to overcome is **offline navigation**. With that there are a lot of moving parts. We will need to calibrate the phone to accurately measure direction and distance so that a user will be able to navigate to a precise location offline for sometime. Since the phone is the only device that will be assumed to be available to the user, it is going to have to be accurately navigating the user. The user's phone is going to

be relied on for verifying the location of the photos. Accurate directions and distances will be key in offline navigation to the exact location.

Another step in guiding the user to a location is **providing them with a map**. Map data for a small location on google maps is about 100MB. That is not an insignificant amount of data and it can add up space very quickly. On top of that, a user is going to need to be downloading data if they are going offline, so this could be a burden on a cell plan for them. With all of this, we will need to provide an efficient way to download offline maps that is storage size friendly. Getting this data to the user is a massive step in helping the user get to a verified location so that they can actually use our app and help with the overall goal of verifying GEDI data. The whole point of citizen science is to make sure that everyone can be involved, so we need to overcome potential storage and download issues for them.

After a user takes a photo, we will need a way to verify that they are in the **correct location** and that they did not upload a photo from a different location. We will need a way to identify precise location/coordinates of the user so that we can attach these to photos for verification. This will be crucial in GEDI data verification as the whole point is knowing which cluster photos are verifying.

We will need to **convert GEDI satellite locations** into coordinate data and then any other data the client asks us to validate. This challenge has to take into account the kind of data we are trying to retrieve, and what we can do with the data. Some of the question we should ask are: Is this data purely numbers? How large is the data? Does it take the whole database into consideration?

Ensure that users with little technological experience can **use our app with ease**, as we plan on deploying our app to rural areas with medium to little exposure to technology, especially with navigation. This is more of a UI/UX challenge. What constitutes a positive user experience? Perhaps we cannot provide too much information on the screen at once, and instead have to guide them page-by-page on how they are supposed to use our app, and alert them of any major changes that will occur if they were to click on a given button on the screen.

We will need to ensure that any integrations of RESTful APIs are **not taxing on the users device** and integrate well with our app. Since the APIs will be dealing with possibly Gigabytes of data, we will have to put a limit on how many times the API gets called by the app. We also have to be very careful with any network related issues, as this app will be used in remote areas with limited network access, making the API calls slower and more demanding than expected.

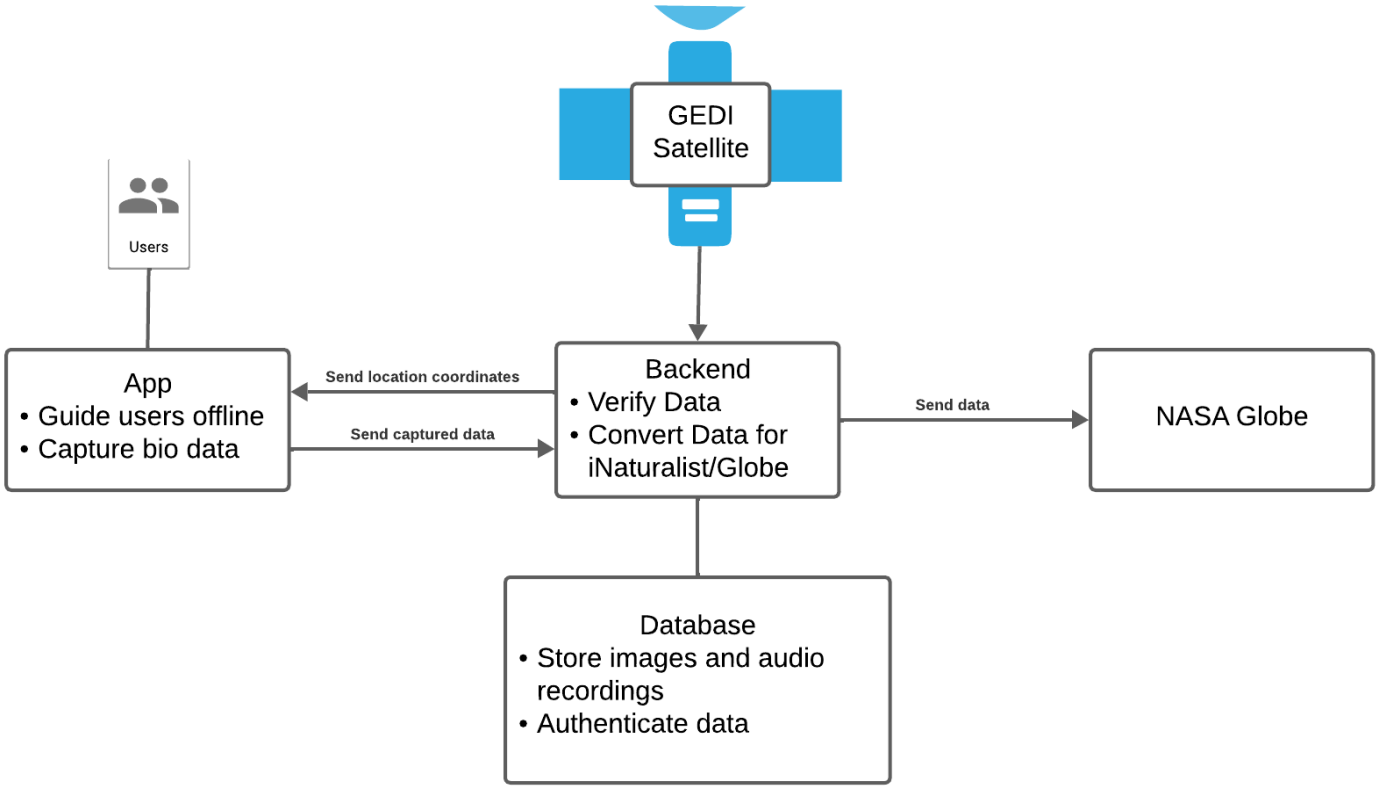
Many of our users will be in the middle of nowhere for extended periods of time but will still be collecting data. We will need to find a way for users to **store data while offline** and then upload verified and accurate data when they are in a location with service again.

We will need to ensure that our app works on **older versions of Android** that our users may have. A lot of users will have older phones with different hardware and software. It is important that we find a way that most of these older devices will still work with our app.

3. Technology Analysis

3.1 Introduction

Our main challenge is collecting and verifying data collected from the GEDI locations. We need to have our users be able to collect data offline while being able to get their location and the data they collect. We have researched an API for getting the map data which is the main component of our product. Our main functions will be for gathering location data, verifying that it has not been tampered with, saving it for offline capabilities, and giving our users access to what areas need data collection. All of these parts should come together seamlessly in the final product with minimal effort to the user.



3.2 Desired Characteristics

Our ideal solution will have to **work on old phones** and primarily on android since that is widespread with our expected users. It will also have to be **cheap to upkeep**

since we are not expecting a revenue stream. We will need **offline functionality** since we are expecting our users to be in the middle of nowhere for much of the time they are using the app. A **map will be available** on the app showing **GEDI coordinates** and the user's location. The app will also have to **verify the location data** since the main goal of the app is to collect data that is not forged or faked in any way. We will also be **uploading the data** to a well known citizen science app so that it is available for widespread use.

3.3 Alternatives

To address some of the location based challenges, we have researched various APIs, one of them being **Mapbox**. Mapbox is a company that developed various APIs that help with mapping and navigation. This came up through the advice of other developers on reddit. Some companies such as AllTrails and Strava use these which is very appealing as these companies heavily rely on route tracking and offline capabilities. Mapbox also had a lot of features that we were not expecting to, but we will go over that more in the next section.

An alternative to mapbox would be an open-source software called **Leaflet**. This software is very simple to use according to online sources. Companies like Facebook, Pinterest and the Washington Post are listed as companies that use this package in their own development. It has been around for the past 15 years and continues to keep itself relevant as a leading map display and navigation tool.

A RESTful API called **Retrofit** is a simple, type-safe API that can convert HTTP API into a Java interface. Developed by Square, the same developers of Dagger and Okhttp, Retrofit is a popular API for Android and Java applications and has been used in a variety of applications because of its efficiency and simplicity. It is a very popular library and will be heavily considered in our project.

Ktor could serve as a very good alternative to Retrofit. Created by JetBrains, it aims to simplify web development in the Kotlin framework by having the API built in the Kotlin libraries. It can build everything from simple services to complex, full-stack applications. It can be used for Mobile back-end services, maintaining and sharing code across multiple platforms. It can also be used to process and store data into the Ktor-powered servers.

Firebase is an app development platform with a variety of features to help you build apps. It has app testing to let you test on multiple devices and to help rollout features. It is very useful for backend features and protection from unauthorized users. Firebase helps you implement signing into your app and it has a lot of other helpful features like monetization, hosting, and databases.

Backendless is a good alternative to Firebase. Like Firebase, it helps you develop apps and is easily scalable. It has database management, ui building and user management. Backendless helps you build apps on different platforms and has the same overall goal

as Firebase. Both development platforms are affordable, scalable and have similar features. The main difference is preference, one could be more useful depending on what you need it for.

Amazon Location Service is a service that lets developers add location and geospatial data to their apps. There are a few different features that could be useful to us. It has map and tracking features, both of which are very important to the project. It also has geofencing which if implemented correctly could help with our location features.

Radar is another geolocation platform which is very similar to Amazon Location Service. It also has geofencing and map features but claims to be more developer friendly. Additionally, it also has fraud detection which would be useful for our project. Radar is proven by thousands of companies that use it worldwide.

Now that we have introduced some of our technologies, we can make a further analysis of them. The next section will help us understand what each one offers and to what extend

3.4 Analysis

Starting off, the Mapbox API has many useful features such as offline capabilities and multilingual maps. These multilingual maps are useful as one of our stretch goals is creating this application in French and other languages. The offline maps allow the user to view where they are when they are in remote locations, although they will need to download this. We as developers will also be responsible for actually tracking them offline. Mapbox provides access to detailed maps in rural and remote areas where there may be trails.

On the other hand, leaflet, which is an open source project, has a lot of contributions to many of their APIs. There is a large library of APIs and the ones that would be most important to the project would be the routing and geolocation. Some of the geolocation APIs are able to pinpoint coordinates which is good, although the routing to remote locations is not the best. It is difficult to implement. There is a library on github to download leaflet maps offline. It is difficult to tell how detailed these maps are for rural terrain and the github code for this project seems to have some pretty big issues (here) although the group has not been able to test it out yet.

Taking a look at a RESTful API that we will be using to deliver and receive data, one API that we can use is the Retrofit API for Android. This API simplifies the process of GETs and Responses by providing a clear, and safe way of defining RESTful APIs. It allows us to make requests and handle responses in an easy and efficient way, which will definitely help us in our communication with other apps such as iNaturalist. There are a variety of ways to use this API which we will cover in the analysis section.

Looking at its alternative, Ktor, this API is built entirely into Kotlin, making its features far more accessible. It can also support both server-sided and client-sided development,

unlike its counterpart that is specifically designed as an HTTP library and can only be used to consume Rest APIs. Ktor can also support WebSocket out-of-the-box to allow for live updates of data. If our application requires high concurrency and asynchronous processing, then Ktor serves as an excellent alternative to fetching Rest APIs over Retrofit.

Figure 1.

	Routing	Map Display	Offline Capabilities	Multilingual	Off street routing (trails)
Mapbox	Yes	Yes	Yes	Yes	Yes
Leaflet	Yes	Yes	N/A	No	No

Figure 2.

	Asynchronous Support	WebSocket Support	Server-Side Capabilities	GEDI Data Extraction	Taxation on Device
Ktor	Yes	Yes	Yes	Yes	Heavy
Retrofit	No	No	No	Yes	Light

The thorough analysis above is going to help the final decision making below as the chosen approach is discussed. This is where the technology will be decided based off the challenges and analysis

3.5 Chosen Approach

To decide the final technology to be used for the app, it is important to take a look back at the initial challenges that need to be overcome. To make the application successful a big part of the application is clearly the mapping. Back to the earlier challenges, providing a user with a map so that they can navigate offline through trails is going to sum up what was discussed previously. Below in Figure 3. Leaflet and Mapbox are broken down to see what each meets.

The transfer of data was a part of another set of challenges that need to be overcome since this application is going to be deployed to older versions of android and users that may be taxed by high data usage. Efficiency will be key. The problem is that high level GEDI data will need to be pulled and getting this information to our app is no small feat. Below Ktor and Retrofit are compared with key factors that are important to the extraction and transfer of the GEDI data.

Figure 3.

Alternatives	Desired Characteristics
Mapbox	✅ Off Street Routing ✅ Map display ✅ Offline mode
Leaflet	🟥 Off Street Routing ✅ Map display 🟥 Offline mode
Retrofit	🟥 Multi-platform compatibility 🟥 Asynchronous and real-time support 🟥 Back-end server support ✅ lightweight resource intensity
Ktor	✅ Multi-platform compatibility ✅ Asynchronous and real-time support ✅ Back-end server support 🟥 lightweight resource intensity

The table above shows that when it comes to the integration of a map functionality into our app, Mapbox contains one extra feature that makes it outshine Leaflet, that being offline functionality. Users of our app will not have consistent internet connectivity, and Mapbox having offline functionality solidifies it as our main API for handling map functionalities.

For extracting NASA GEDI data, Ktor is the clear choice. While Retrofit may be lighter on older Android devices, Ktor offers superior flexibility across platforms and enables real-time data authentication. Additionally, Ktor's backend server support allows us to streamline development between our database server and the app.

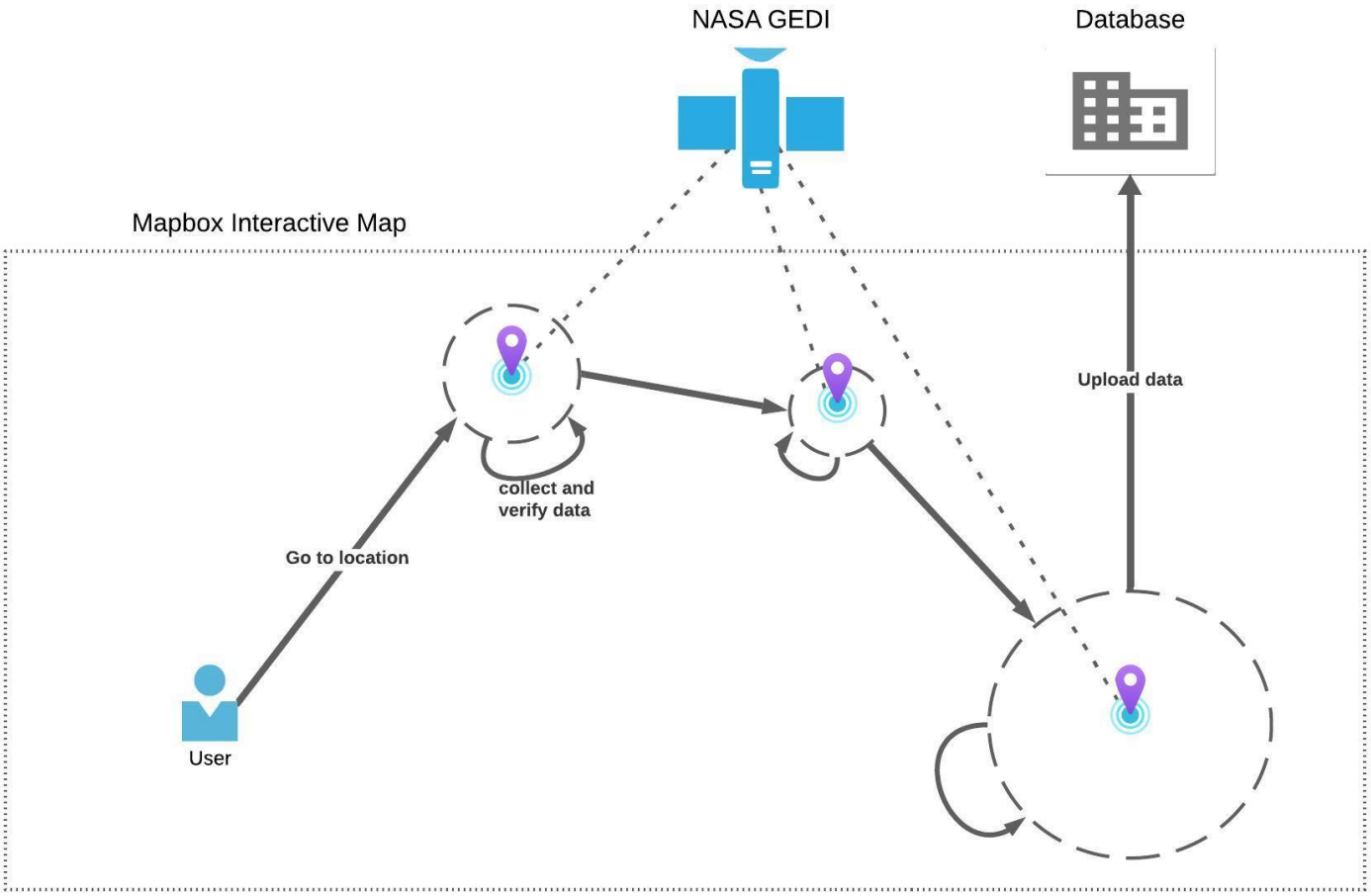
3.6 Proving Feasibility

Many location and mapping features will be integrated into our app. Mapbox API specifically allows the integration of a fully customizable and interactive map that can contain data on geolocational points as well as regions of interest. Combining these features with the coordinates of carbon footprints and other relevant data that we can extract from NASA GEDI using Ktor and implement them into our integrated Mapbox interactive map, we can assist in making it easier for users to not only know exactly

where they are but also know where to go to retrieve data for our collection. This idea will also support our ability to verify and check that the collected data was retrieved from the same coordinates given by NASA GEDI.

In retrospective of the proposed idea, we will test our approach with the integration of an interactive and customizable map. We will then customize that map to show the user's current location coordinates as well as coordinates of regions of interest.

1. After having the user move to a specified location on the map, the demo will then ask the user to take an image of a specified animal or plant.
2. Once a user tries to submit the image, the app will go through a fraud check by ensuring the coordinates match.
3. Once that is verified, the demo reaches a conclusion and the image is added to our database.



4. Technology Integration

After examining the technologies that are available to us, we have come up with an architecture that we can benefit from the most. Although there may not be technologies that exist to handle our exact needs, they all provide some way for us to get a good foundation in different pieces of our application. Luckily our architecture allows for us to be modular in our design. The way our technologies will be integrated should be extremely smooth because of this. Below we will discuss different parts of our architecture and how the technologies will get pieced together.

To start we can walk through what a user experiences so that it is clear when a certain technology would come into play. A user will first open our application and that is when they will see the UI. In this UI, it will have been made by the Android development tool, Kotlin. This will be the foundation for which the rest of our technologies can be applied since Kotlin is how we develop and build our application to be user ready.

When a user is ready to find a GEDI footprint near them then we will need to call on the mapbox API. Using the mapbox API we should be able to locate where the user is, and based off of a certain radius, pull the nearest coordinates for the GEDI footprints in this area. Once we have queried the coordinates from our database, we will be able to plot these again using Mapbox API so a user has a choice of where to go. Mapbox has both web based and javascript capabilities, so we may be using Retrofit to handle the requests we make to mapbox. The purpose of that is to handle data in a more efficient manner depending on what we discover through greater implementation.

The next part of our application will involve a user navigating to a location offline. Again we will be using mapbox API for routing if available and guide users to the location when they are online. When they are offline, that is when we will be able to download the app, but that is when we will also need to help them navigate to a precise location offline. As of yet we have not found anything that can explicitly navigate offline, so this is where much of our own code might come in to utilize sensors already available to us on the phone so that we can accurately track a user offline to a GEDI footprint.

After a user is in the footprint they will begin being able to take photos and record audio. Our group is not handling the gathering of photos, so we have not explicitly researched anything to assist with this. We are also going to end up potentially needing to navigate a user back to their starting location if they are offline.

Our integration will be smooth and anything that we need is functioning separately in the sense that they do not need to be configured a certain way to work. Like I said, they are modular, so as long as they are supported by Kotlin, the APIs will be usable separately as long as the data returned is usable.

5. Conclusion

The main motivation behind our project is assisting communities in areas affected by conservation efforts. Our plan is to have them participate in citizen science research through the use of our application which allows them to collect valuable data for the NASA GEDI project. Our mobile application will consist of features such as an interactive map, possibly usable when offline, that shows the users regions of interest within their area through the use of technologies and APIs such as Mapbox andKtor, and allow them to collect data in those points of interest. Some of the challenges we are expecting during the implementation of our project include, but are not limited to, offline functionality, storage feasibility, and backwards compatibility with older versions of Android. If we can get through those challenges, then we are confident in our ability to implement a solution that benefits both the researchers in their data collection and the users in supporting their local livelihoods.