Technological Feasibility Analysis

4 April 2025

ArchaeoSight

Sponsor: Dr. Kayeleigh Sharp

Mentor: Jeevana Swaroop Kalapala

Team Members: Larry Griffith, Dazzion Porter, James Harris, Levi Ly

Table of Contents

. . .

Introduction

Most of the land in the world has been discovered and documented. As time has gone on, people have discovered new lands and moved on from old ones. Archaeology and anthropology aim to learn about the lands they left behind as well as the people who left them. Modern archaeology uses aerial photos, lidar, GPR (Ground-Penetrating Radar), and texts to figure out where artifacts and important sites may be hidden. Our client, Dr. Sharp, is helping conduct digs in Huaca Letrada on the north coast of Peru to try and find evidence of human activity. Her focus is to investigate the spatial and social contexts of certain activities, primarily copper procurement, metalworking and pottery production. She does this by collecting pXRF (x-ray fluorescence spectrometer) data, which tells her the composition of the soil. The issue with this approach is the fact that 5 mm in diameter of soil can be tested at a time.

While these modern methods allow archaeologists to collect incredibly precise data, they also introduce new technological challenges—particularly when it comes to organizing and presenting that data in a meaningful way. The limited scope of each individual reading, especially with pXRF tools, means that a large amount of complex, fine-grained data must be collected, transferred, and analyzed in tandem with GPS and image-based data. To address this, our team is building an app that not only consolidates this data but also uses machine learning to help guide archaeologists toward potential new discovery sites.

Technological Challenges

There are several major design decisions we need to make in the development of this project. Before getting into more technical details, we have a number of requirements for our project. We must have a way of passing data from the specialized tools our client and her teams use, which includes SciAps and Bruker XRF spectrometers, to our app. Our app should also accept image and GPS data. Using all of this information, our app will have to use a machine learning framework to estimate potential sites for more sherds and other archaeological fragments. Early on, we will need to settle on an integrated development environment (IDE), a framework, and a programming language. Additionally, we should consider what kind of database and management system we will be using. In developing our app, we must provide users with a responsive and efficient experience to minimize frustration in inputting data. Our final consideration regards how internet connectivity should work for the app, as a good connection cannot be guaranteed for our users in the field.

Technology Analysis

Section 1: Cross-Platform Mobile Application

i. Intro

The project requires a cross-platform mobile-application that enables archaeologists to collect, process, and analyze data from multiple sources, including pXRF readings, GPS coordinates, and images. The app must also include a ML (machine learning) model that can predict areas of archaeological activity. One of the major challenges is to ensure that the application works seamlessly across multiple devices (phones, tablets) from multiple manufacturers (iPhone, Android). Additionally, the ML algorithm needs to give quick and accurate responses.

ii. Desired characteristics

The application has to be able to be accessible and consistent from both iOS and Android devices, give a quick and accurate analysis of the data given to the ML algorithm, and be maintainable and scalable for the future. Luckily, there are two great options that fulfill these requirements.

iii. Alternatives

The first of which is React Native. React Native uses JavaScript and React to build multi-platform mobile applications. It provides near-native performance using a JavaScript bridge, allowing it to work on iOS and Android. Importantly, it has a massive developer community and extensive third-party library support.

Another option is Flutter. Flutter uses the Dart programming language which compiles directly to native ARM machine code, which gives it a huge performance advantage. Dart is "write once, use everywhere", meaning it can seamlessly work on iOS and Android devices. It has a large developer community and is backed by Google.

iv. Analysis

To pick the right one, we need to look at what each framework offers in relation to our needs.

Requirement	**Flutter**	**React Native**
Cross-Platform Capability	✅ Yes	✅ Yes
Performance and Responsiveness	✅ High	⚠️ Moderate
Ease of Development/Use	✅ High	✅ High
ML Integration	✅ Strong Support	⚠️ Moderate
Open-Source	✅ Free	✅ Free
UI/UX Consistency	✅ Excellent	⚠️ Dependent on Native Modules
Ecosystem	✅ Strong	✅ Strong

v. Chosen approach

We are going to choose Flutter as our app building toolkit for a few reasons. The first of which is performance. Because Dart compiles directly to native ARM machine code, it will provide overall better performance over JavaScript, which must implement a bridge to the environment it is working in. The second reason we are choosing Flutter is its machine learning integration. Dart allows us to use TensorFlow Lite within mobile applications to make the ML algorithm development easier. It also has access to flutter_map and google_maps_flutter for GIS and GPS integration. The third reason we chose Flutter is because of the UI/UX consistency. React Native requires different modules for iOS and Android, whereas Dart does not.

vi. Proving feasibility

To validate the choice of Flutter for the cross-platform archaeological analysis application, we propose a series of tests and demos. First, we will develop a basic prototype that integrates GPS data collection and image capturing to assess the app's core functionality. Next, we will implement a basic TensorFlow Lite model within Flutter to evaluate its ability to handle real-time data processing for machine learning tasks. Additionally, we will integrate a simple GIS visualization component to test compatibility with spatial data and ensure smooth handling of geographic information.

Section 2: Data Display Scope: Balancing Visualization and Feasibility

i. Intro

One of the key technological challenges of this project involves determining the scope of data visualization presented to the user in the interface. Since our application aggregates high-resolution 3D imaging, pXRF data, and GPS location data to assist archaeologists in identifying areas of interest, we must make a critical design decision: Should the app display all incoming data streams in a fully formatted, user-friendly interface, or should it prioritize formatted views for system-generated recommendations while leaving raw data in a standard, less-polished format? This decision significantly affects development time, user experience, and feasibility in the field.

ii. Desired characteristics

To evaluate this challenge, we consider several essential characteristics: Development Cost and Time – fully formatting all data inputs would require significantly more UI/UX development resources; Usability – archaeologists in the field need a quick and clear display of recommended locations and supporting data, not necessarily exhaustive visualizations of raw input; Maintainability – a simpler standard display for raw data minimizes future update costs; and Device Compatibility – the solution must run efficiently on smartphones and tablets in remote settings.

iii. Alternatives

We considered two main approaches. Alternative A is a full visualization model: All data—3D models, pXRF readouts, and 3D imaging data—are presented in a detailed and polished UI. Alternative B is a hybrid approach: key predictions and outputs are well-formatted and highlighted, while raw data is viewable in a simple, tabular or minimal form. The full visualization model has been explored in commercial GIS apps such as ESRI's ArcGIS Field Maps, but these often require high-spec devices and extensive rendering tools. The hybrid model has been used in lightweight academic tools, such as survey companion apps in archaeology, that focus on clarity and efficiency over visual richness.

iv. Analysis

On performance, Alternative B clearly outperforms A by reducing device load. Development time is also shorter for B, as fewer UI components need refinement. From a usability perspective, archaeologists benefit most from easy-to-interpret predictions rather than navigating dense data. Maintainability is another strength of B, since the core interface remains stable while raw data formats can change over time. Compatibility also favors B, as it is less demanding on field devices with limited processing power.

v. Chosen approach

After comparing both alternatives across our desired characteristics, we have not yet made a final decision. While the hybrid display model (Alternative B) currently appears to offer a balanced and scalable solution—highlighting ML predictions in a clear format while still allowing access to raw data—we plan to consult further with our client to better understand their expectations and preferences regarding data visualization. Their input will help determine whether the added complexity of a fully formatted display is necessary for the intended users or if a simpler, hybrid approach will meet their needs effectively.

vi. Proving feasibility

To ensure that our eventual decision aligns with both technical feasibility and user needs, we plan to develop early prototypes that showcase both data display approaches: a fully formatted interface and a simpler hybrid layout. These prototypes will allow us to test performance on mobile devices, evaluate clarity for archaeologists in the field, and gather direct feedback from our client. This hands-on evaluation will help confirm which approach best balances usability, performance, and development scope, ensuring the chosen solution supports the real-world conditions and expectations of the project.

Section 3: Integration of Client Tools and Interfacing with Hardware

i. Intro

To give informed suggestions of locations of high anthropogenic activity, it is important to first understand where our data is coming from and how it is collected. Our client Dr. Sharp plans on using ArchaeoSight in the field, in conjunction with XRF analysis tools such as the SciAps X-200, Bruker Titan 800, and Bruker Tracer 3. These are all tools for safe environmental soil analysis, capable of precise alloy identification using X-ray fluorescence technology. By combining the geographic visualization of ArchaeoSight and the precise elemental identification of the XRF tools, Dr. Sharp will be able to conduct thorough field studies and will be able to efficiently identify anthropogenic hotspots.

ii. Desired characteristics

To ensure that our ML algorithm is making accurate predictions for areas of high-level geochemical production, one of our main objectives is to streamline spatial data with the information found by the XRF tools. This integration will enhance the algorithm's ability to identify key geochemical patterns, allowing for a deeper understanding of the relationship between elemental concentrations and its possible distribution. The given precise field data and validation of results against field observations should improve the reliability of the algorithm's predictions.

iii. Alternatives

The pXRF tools capture elemental composition data in real time, recording the concentrations of the elements detected. That data is then stored, formatted, and then exported as CSV, PDF, and PDZ files; offering connectivity options through Wi-Fi, Bluetooth, or USB for file transfer. The versatility of these file formats could allow for seamless automated data transfer.

iv. Analysis

Depending on which pXRF tools are used, the collected data can be formatted into a few different options. Exporting CSV files could prove to be advantageous, due to its seemingly universal compatibility and is supported by most data analysis software. It's flexible, allowing for the ML algorithm to easily harvest data; however, depending on how the data is structured, it could be a challenge to interpret. The option of exporting data as a PDF could be easier for readability but is not very suitable for integration into software for further analysis. The PDZ file option holds rich data and is optimized for use of the manufacturer's proprietary software. This option would be perfect for our intended use but has very limited compatibility as it requires very specific software to be properly utilized.

v. Chosen approach

With the current information given, the approach with the best outlook is exporting the data found by the pXRF tools as CSV files, as the data will be easy to clean, read, and manipulate. Although further interaction with these devices is needed to make an official choice.

vi. Proving feasibility

As stated before, further interaction with the pXRF tools is needed before an official decision. Exporting data as CSV files could prove to be the most convenient and available option, allowing for easy integration into ML software.

Section 4: Offline and Online Support

i. Intro

Archaeological sites are not typically equipped with good, stable internet. As such, an app that requires a constant connection to the internet will be inefficient at best and unusable at worst. However, a completely offline app will not meet our needs either, as our app should be able to send and receive data from a database.

ii. Desired characteristics

Our final solution should allow our app to function offline and take advantage of an internet connection while online. Offline, our app can accept and store data locally, and it may identify potential dig sites based on the local data. Online, the app can send data to and download data from a database. It should also be able to make estimates using both the local and online data. Additionally, the device going from online to offline or vice versa should minimally interrupt the user's experience; the app should not have to refresh or reset. This type of app is often referred to as an "offline-first" app.

iii. Alternatives

There are three main parts of an offline-first app, and each has different potential approaches that can be taken.

The first is how data is read. It is important to the user that data can be accessed as quickly as possible without relying on the internet. One approach is using local data as a fallback for when the user is offline. If the app fails to obtain a piece of data from the remote API, then it instead returns locally stored data. A second approach is to use a stream, which first emits locally stored data to the user before attempting a network call. Only when the data is not found locally is a network call made. A final potential approach is using only local data. The app will simply sync with the database either manually or whenever the device is online, and the app will only use the downloaded data.

The second is how data is written. The first approach would be to only allow writing data when online. The next approach fully embraces the offline-first architecture, where it first stores data locally before attempting to send it to the API service.

Finally, we consider how data is synchronized. We could approach this by simply creating a timer that runs every 10 minutes, for example, and attempts to sync the data. This could also be made to run in the background, which avoids requiring the app to be open. An alternative approach would be to utilize a push service. The server would notify the app when any data has changed, rather than having a synchronization task running in the background.

iv. Analysis

Because these are quite complex to test, we did not write code ourselves to experiment. Instead, we researched each method extensively to determine which approach for each layer best suited our needs.

v. Chosen approach

For reading data, having local data as a fallback is a simple approach, but attempting to make a network call first would take some time before resorting to local data. Using a stream is generally faster and less prone to error, though it is more complex to implement. Finally, relying only on local data would be fastest to retrieve data and be easiest to achieve, but it would require storing large amounts of data locally, which would likely be inconvenient for most users.

Requirement	**Local Data Fallback**	**Stream**	**Local Data Only**
Ease of Development	✅ High	⚠️ Moderate	✅ High
Robustness (Error Resilience)	⚠️ Moderate	✅ High	❌ Relies on one source of data
Speed	❌ Always makes network call	⚠️ Quick for local data	✅ High, given local data
Device Impact	✅ Low	✅ Low	❌ Heavy on storage

With these factors weighed, we will opt for using a stream. Despite its higher complexity, we believe it will be best for handling the large amounts of data our app will require.

For writing data, online-only writing ensures that all data is fully synchronized. While users will still be able to read data offline, they will be unable to write any data. This is not suitable for our needs, as our users will often lack an internet connection while out in the field, which is where data will usually be written. The second approach allows writing data while offline, letting it be stored locally. The drawback to this is that if the app is unable to send the data to the API service, the local database and the API service will be out of sync.

Requirement	**Online-Only**	**Offline-First**
Robustness (Error Resilience)	✅ High	⚠️ Synchronization conflicts may arise
Flexibility	❌ Inhibits data collection in the field	✅ High

For our purposes, offline-first writing is the only option that fits our requirements. Disallowing writing without internet defeats most of the purpose of our app.

For syncing data, a timer would be simple to implement, but it can be quite taxing on the device, degrading performance and reducing battery life.
Unnecessary battery draining is especially hindering for our users, as they may not have convenient access to chargers for their devices in the field. In addition, some devices outright limit the number and power draw of processes that can run in the background, creating an additional obstacle that we will have to work around. On the other hand, pushing data from a server would only notify the app when data needs to be updated, avoiding the need for a background process. This approach will likely require additional services such as Firebase for messaging, increasing the complexity of the syncing process.

Requirement	**On a Timer**	**Pushing Server Data**
Ease of Development	✅ High	⚠️ Requires additional libraries and tools
Reliability	⚠️ Depends on sync interval	✅ Syncs as soon as possible
Device Impact	❌ Strong impact on performance and battery	✅ No background process required

After looking into our options, we have decided that pushing server data will be a more reasonable approach. The impact on the device from syncing on a timer is too much for our users and is the biggest factor in our decision.

vi. Proving feasibility

Once we become adapted to our chosen framework, we plan to develop a basic prototype that combines and utilizes all three chosen approaches for reading, writing, and synchronizing simple data. We will test this prototype both online and offline to ensure that it works as expected.

Technology Integration

Take photos -> upload to DB and ask model -> upload pXRF data -> upload to DB and ask model

With the individual challenges addressed, the next step is to integrate the various solutions into a unified and efficient system capable of meeting the project's overall requirements. To achieve this, we need to design an architecture that seamlessly combines the chosen technologies—Flutter for cross-platform development, machine learning with TensorFlow Lite, GPS and GIS integration, offline-first capabilities, and integration with pXRF tools—into a cohesive application. The system must ensure smooth data flow between components, maintaining both functionality and performance across diverse devices, from data collection in the field to analysis and visualization. Data collected from pXRF devices and GPS will be processed locally or through the ML model running in the Flutter app. This processed data will be stored locally in an offline-first model and, when possible, synced with the remote database using push notifications for real-time updates. The system's architecture will ensure the app remains responsive, even without constant internet access, by relying on a stream-based approach for data retrieval and allowing offline data entry. The machine learning model will integrate into the app to provide archaeologists with predictive insights based on data collected in the field, while the GIS system will provide spatial data visualization, crucial for identifying potential archaeological hotspots. Through this integrated approach, the app will deliver an intuitive user experience while ensuring high performance, accurate analysis, and scalability for future requirements.

Conclusion

In modern archaeological practices, precise and cutting-edge technology such as the pXRF tools are used. But these tools on their own call for manual data recording which can make fieldwork a bit tedious for the client. The purpose of ArchaeoSight is to uptake the heavy lifting and do the guesswork for you by creating a platform that streamlines and visualizes data, allows for possible collaboration, and uses ML for making accurate and reliable predictions of anthropogenic hotspots.

The convenience lies within the projected compatibility of ArchaeoSight, set to be available for Android and iOS devices. ArchaeoSight also offers portability with its intended offline capabilities. Our aim is to create a seamless UI fitted for archaeologists around the world, with a focus on hotspot predictions by ML algorithms, and the accessibility of clear and raw data.

These innovations together not only enhance efficiency but also reduces the workload for archaeologists, allowing them to concentrate on interpretation and discovery rather than manual data management. By bridging AI technology with archaeological field expertise, ArchaeoSight aims to be an indispensable tool in the future of Archaeology.