MoGreen

Technological Feasibility

Version 2

November 7, 2018

Project Sponsor:
Ellen Vaughan and Brock Brothers

Team Mentor:
Ana Paula Chaves Steinmacher

Team Members:
Cassie Graham, Jennie Ryckman, Chase Mosteller, and Justin Shaner

NAU NORTHERN ARIZONA UNIVERSITY

Table of Contents

1. Introduction

Surrounded by a ponderosa pine forest, the NAU campus strives to be as green and sustainable as possible. NAU has glass, paper, plastic, and battery recycling located throughout the campus. As NAU continues to grow its student population, instances of overflowing trash bins, full dumpsters, and campus litter rises as well. Basic maintenance needs occur more often as sprinkler heads are accidently broken by passersby and other minor issues.

This has been identified by Ellen Vaughan and Brock Brothers who lead the efforts for sustainability on the NAU campus. Ms. Vaughan was the manager of the Office of Sustainability here at NAU. Mr. Brothers is a supervisor and coordinator for NAU's Moving and Recycling. Everyday there are issues Mr. Brothers and his team deal with.

Currently, there are three main types of trash containers around NAU. There are the standard concrete types and the large dumpsters. The third type is the "Big Belly" trash bins and there are over 300 of those across our campus. These trash bins are very useful since they are connected to a network and can send out notifications when full. They are also solar powered and can compact the garbage they collect. However, there are situations where the bin entrance can be blocked. The system is limited and cannot report such obstructions. Mr. Brothers and his team will not know if there is an issue unless they come across it and visually see the blockage or someone reports it. NAU's Office of Sustainability also works with repairing building fixtures, general building maintenance, landscaping, and creating work orders if issues are found when general inspections are made.

The current way for a student to report minor issues is to call NAU Facilities or report it using their website. Nearly everything done in the effort to keep the campus clean and maintained is completed manually. There is not a system in place to accurately track when areas around NAU have been cleaned. There are not many tools for the NAU community to get involved in helping. They are limited to calling or reporting via a work request made online. Either way, the reporting method is a little outdated and most students will not make the time to report minor issues.

The name of our project is "Clean My Campus" and it will be a mobile application. This application will be developed by our four-person team. Team MoGreen looks forward to working on this application and embraces all challenges that may emerge during its development. The mobile application proposed will make reporting minor issues easier

and allow NAU facilities to receive and track these reports faster. This application will help NAU's sustainability and keep our campus clean and green.
The solution we are designing is a mobile application that will be able to designate sectors throughout the campus. We can use these sectors to show visually when an area has been cleaned. We can create a reporting method to streamline reports of minor issues. We can use the reports to generate data which can track high volume areas. We could also use this data to help NAU green groups get organized quickly and clean the campus more efficiently. Ideally, this application will make it easier for the NAU community to get more involved. Minor issues like overflowing trash cans or full dumpsters should not require more than opening the application and sending out a report.

Ms. Vaughan is interested in having a mobile application developed which can help sustainability, and not just for NAU. She envisions that NAU will be the pilot campus where the initial application is developed. If successful, she will bring it to other campuses in California. Mr. Brothers is interested in the application for its use in helping he and his team maintain the campus in general. He also would like to have the application help identify trends and areas of high trash volume. Creating a sustainable campus is everyone's responsibility and this application will contribute to that effort.

This document covers the technologies involved in creating our application. We will be looking at multiple software that will implement what is needed for development. The software selected will be performing the tasks we believe are feasible for our project's scope and timeframe. The topics we will cover in this document are the mobile application language, database solution, web hosting service, and the mapping application program interface (API). These topics are the main components which make up the minimum product for our clients. The software chosen to solve the problem for each topic will be shown why it is the best choice for our development. The outcome of this document is to have a thorough record of our team's reasoned explanation on why we selected the specific software for this application's development.

2. Technological Challenges

Our minimum viable product (MVP) has four main elements that we will need to consider. For our MVP, we will need to implement the following:

- Mobile application development environment.
- Database Management System (DBMS).

- Mapping API.
- Web Hosting Services.

The language or environment we choose to write the application in will inform the platform of the application, and the potential for cross-compatibility. We also need to consider what languages our team members have familiarity with or provide ease of use as this will be the crux of the project. Our main consideration here will be platform and language compatibility with whichever API we choose for our graphical user interface (GUI) elements.

Choosing a database system is likely one of the more significant of the technological challenges. Our main consideration will be familiarity and ease of use. We will need to designate an administrator account, which will have permissions for handling reports and viewing statistics on the website.

The mapping API we choose is integral to our project. We need to choose an API that allows us to divide the map into sections and color code the sections in a variety of ways. It is preferable that these functions are built in to whichever API we select.

Choosing website technologies is perhaps the easiest component for our MVP. We are all familiar enough with website implementations that any we choose will suit our purposes well enough, especially given that the website is meant to serve as a backend that stores data and provides statistics.

The main interface of this project is the application itself. The website will see limited use as it is meant to be mainly for campus administrators. Our main concerns regarding the website are the interactions between the application, database, and website itself. Our website will provide statistics in possibly different formats that are collected through the application. We will need to figure out a way to collect and store that data and have it accessible from the website.

3. Potential Expansions

Features for our “comfortably equipped” solution or stretch goals do not necessarily require additional technologies. Our primary stretch goals are:

- New campus creation and administration.
- Gamification.

- Map icons and interfacing.
- Geolocation features.

Allowing the creation of new campuses would require us to have multiple administrators and handle permissions for each. This way each administrator would only have the appropriate permissions for the campus they handle. This would affect our database management.

Implementing gamification features would allow users to form groups and earn points individually or in their groups. This would affect our databasing and the type of data and statistics the application will collect. We would need to extend it to attributing reports to users and assigning some point values. We may also need to create a competitive element such as a 'high score' system to stimulate participation. Users should also be able to view their team on the application.

Our application could potentially show icons to the user that tell them where trash and recycling resources on campus are located and what type of waste they process. It should not be too difficult to implement this but still falls under potential expansion as it was not deemed necessary. Implementation would interact with the mapping API we choose. We may want to have this be an optional filter on the map GUI, so users can choose to view it or not.

Geolocation is the most complex potential expansion. Geolocation within this application would allow the map on the application to open directly to where the user is at that time. It would also allow problems reported to be tagged with an exact location instead of the predetermined zone. Implementing this may require use of a separate API in conjunction with our mapping API.

4. Technology Analysis

This section will analyze different software solutions that the components of our minimal viable product will be implemented with. For each of the MVP components, we will give a brief introduction of what the component is trying to address for development. We will then discuss different software solutions, compare them based on appropriate criteria, and name our decision. Each section has a table that shows how each solution compares to another and a summary of how we will test the solutions feasibility.

4.1 Mobile Application Development Environment

The development environment and languages we choose will determine how our application works and what platforms it is available on. Our team decided that we prefer our application to be cross-compatible if possible, otherwise be for Android alone. IOS functionality is not a priority for our application at this time since Android development tends to be more intuitive and practical for the programmer. Price is not particularly a factor here, as most environments and frameworks tend to be free.

Our priorities for choosing an environment are:

- What languages are compatible and our familiarity with those languages.
- The learning curve involved with the environment and those languages.
- Integration with our selected maps API.
- Cross-platform implementation.

The development environments under consideration are Android Studio, React Native, and Ionic.

4.1.1 Solutions

1. Android Studio:

   Android Studio is a development environment solely for implementing Android applications. Android Studio does not provide any support for iOS. The primary languages used to develop in Android Studio are Java and XML. Android Studio has built-in integration with the Google Maps API and lots of tools for testing and debugging as a fully-fleshed IDE.

2. React Native:

   React Native is a command-line based framework where users create applications typically using JavaScript and React. Applications can have cross-platform implementation. There is some support for the Google Maps API, but it is lighter than for Android Studio. Applications built in JavaScript or another compatible language can still be opened and emulated using Android Studio or another development environment.

3. Ionic:

Ionic is a framework that uses Angular, CSS, and HTML. Ionic does support cross-compatibility and is less intuitive than React Native. Ionic does not have much support for Google Maps, which will add some overhead when we are creating this map-based application.

4.1.2 Chosen Approach

Currently our team plans to use React Native to develop our application. Using the React Native framework will allow cross-platform deployment regardless of what environment we use to do the programming, although we will make use of the tools that Android Studio offers as an IDE for testing purposes. Even if we do not deploy our application for both iOS and Android, it will still be possible using the React framework for future expansions of this application or with implementation of our stretch goals.

Mobile Application Development Environment Comparison Table					
Scale 1-5 (5 is best)	**Maps Integration**	**Language Compatibility (how many languages, how familiar)**	**Learning Curve**	**Cross -Platform Compatibility**	**Total**
Android Studio	5	4	4	1	**14**
React Native	4	3	3	4	**14**
Ionic	2	2	3	3	**10**

4.1.3 Proving Feasibility

Proving feasibility for a development environment should be straightforward. We should develop a simple application and show that it is usable on an actual cellular device. Since integration with a mapping API is a priority here, we should implement some sort of map GUI within our simple application that can assess the level of support and compare it to our other potential solutions.

4.2 Database Management System

Our team will need a database management system that can receive data from the mobile application. The DBMS we select will need to store string characters, numeric values, and images. The database will hold information for map data points, user data, and images for reporting purposes. Storing images is the largest challenge for our DBMS but each proposed solution can handle the storage and retrieval of them. There are many database systems we can choose from and most of them fall under the two most popular types of databases, SQL and NoSQL. We will be hosting the database locally and will not need to use cloud-based capabilities. We will be looking at each database solution for its data-type storage, ease of use, documentation, and price.

4.2.1 Solutions

1. Microsoft Access:

   Microsoft Access is seen by many as outdated but newer versions of Access can implement a database for our needs. Microsoft is still supporting it and will continue to do so for the foreseeable future. It supports both the creation of databases on a local machine or on Microsoft Azure, a cloud-based solution. Storing floating point values and text is a basic feature but storing images is limited. Access has visual tools for easily viewing database information and it is ready to use out of the box. Documentation for using Access is very detailed since it has been in use for quite some time now. This software has a monthly price plan or can be paid yearly but either way is affordable.

2. MongoDB:

   MongoDB is a database solution that uses NoSQL. It is a free and open-source database software. It is well documented and can host a database on a local machine. They also have cloud-based storage possible as well, but this comes with a price. This product is updated regularly and supported by a dedicated company. This database can store floating points and text as well since that is very basic. Storing images is supported by the software but can cause performance loss.

3. MySQL:

   MySQL has different versions ranging from expensive commercial product to the free, but still highly capable, open-source community version. It can host a

database locally or use cloud-based storage for a price. It is versatile and can be used on nearly every operating system. This software is updated frequently and is supported by Oracle and they will be supporting it for the foreseeable future. Storing floating point values and text is a basic feature. Storing images can be done by storing the images in the database but this will cause performance loss. An alternate way is to store the images in our web server and then make references to the images.

Database Management System Comparison Table					
Scale 1-5 (5 is best)	**Data-Type Storage**	**Ease of use**	**Documentation**	**Price**	**Total**
Microsoft Access	2	2	5	3	**16**
MongoDB	4	3	5	5	**22**
MySQL	4	4	5	5	**23**

4.2.2 Chosen Approach

Our team will be using MySQL for our database needs. It is one of the most used database software world-wide and has been proven time and again for performance and stability. Our needs are simple and the other two options provide a little too much overhead. While MongoDB and MySQL are both free, our team has had more experience using MySQL. Our experience with MySQL and image handling options are what made us choose it over MongoDB. The price for Microsoft Access is unnecessary for our needs. Documentation for each option is abundant as all of them are very popular database software used all around the world. Scalability is not a concern since we will be storing small amounts of data which we can easily be stored on a local machine. This data does not need to be kept forever, a few years at a time at most. The data should be small enough that exporting or copying the data is a non-issue. The database we need to implement is simple and we are confident that a MySQL database is the right solution for this project.

4.2.3 Proving Feasibility

Our primary concern when proving the feasibility of MySQL for our DBMS is the ability to store and retrieve images. This can be shown by taking a picture from our skeleton mobile application and storing the image into our web server. We can then store a reference to the location of the image and show it on our website. We will also need to be able to store map locations and then input them into our map API. The map API uses float values for coordinates. We can store the coordinate values then pull the data from the database whenever we need to display points or outline the regions. The final proof of its feasibility is to create a user registration by applying usernames and a password. We can create a test username and password and verify it.

4.3 Mapping API

Choosing a map API is a critical technological feature of this project, as this application's main features are very dependent on what our map API makes possible for us to do. Our main priorities for choosing a mapping API are functionalities such as:

- Allowing us to define areas and display them accordingly.
- Allowing it to default to an area upon opening.
- Recording a user's location.

The map must allow input for any data that needs to be received while also being able to display information as requested. Three options have been selected and compared, they are Google Maps, Apple Maps, and TomTom Maps API. From these we must determine which map API will allow us to implement the above requirements.

4.3.1 Solutions

1. Google API:

   Starting with Google API, this option allows us to select a static or dynamic section of the map, proving useful if this application were to be used for multiple campuses. The Google API for marking zones has OnClick options which are helpful for creating zones without location features. The API also has options for creating heatmaps which is one of the ideal ways we can express the data we

collect. Along with these, this API can work with any platform we decide to use. The only potential downside here is the cost of using this approach, however with implementing only one campus, as required by our MVP, this may be negligible.

2. Apple Maps API:

   Apple Maps is the map application used on most iPhones in place of Google Maps. While this choice may be somewhat flexible to use on multiple devices, it may not be possible to use depending on our platform choice. Apple Maps allows us to dedicate sections, be they static or dynamic, allows us to dictate zones, and show place markers. A downside of this choice is how new this API is and its limited features. There is limited data logging with Apple Maps which is a considerably large feature of our product. This API has a cost however it can be ignored if the usage is low.

3. TomTom Maps API:

   TomTom Maps API includes many features of Apple Maps but would be compatible with any device we choose to implement compatibility for. Though TomTom Maps is malleable, we still run into some of the same issues with Apple Maps. Features included are generally aimed towards an application for directions. The price is also low as the software is uncommon and has restricted features. TomTom Maps includes necessary features of a map application but does not include a lot of the built-in functionality of the features we will implement in our MVP, nor any for our stretch goals. Having to implement those features manually will add a lot of overhead.

4.3.2 Chosen Approach

Our final decision for this application is Google Maps because if its flexibility and the features included. This solution has built-in functionality for all the features required in our MVP and would allow additional features from our “comfortable solution” to be easily added. Google Maps has an extensive library which would allow more diverse features to be added in the future.

Mapping API Comparison Table					
Scale 1-5 (5 is best)	**Pricing**	**Feature Diversity**	**Flexibility**	**Management**	**Total**
Google Maps API	4	5	5	5	**19**
Apple Maps API	4	3	3	3	**13**
TomTom Maps API	5	2	3	1	**11**

4.3.3 Proving feasibility

Google Maps already includes many features that we will need to utilize. The basic functions of this API will allow us to select an area on the map and display this zone statically or dynamically, allowing us to either include the whole map or decide to use only the areas we need in the static approach. To prove feasibility, we should implement a map in a simple application and show that we can partition it, color-code it, and collect location data.

4.4 Web Hosting Services

Another main challenge is to find a suitable web hosting service for our application's website. The service should be able to support and store our entire website front-end. It should also be able to link to both our chosen database and our application. It must be reliable enough to support these requirements as needed without interruptions to service in the future. Some services we are considering are freehosting.com, NAU's own server hosting, and Heroku. The main issues we are concerned about are storage capacity, database support, availability of features, and price.

4.4.1 Solutions

1. Freehosting.com:

   Freehosting.com is a free web hosting service where you can pay for additional features. Without paying, Freehosting offers 10GB of storage, one MySQL database, and unmetered bandwidth. The unmetered bandwidth should prevent interruptions in service for our website. Databases on Freehosting are limited to MySQL. Even if you pay a subscription you will only get more MySQL databases

as opposed to different SQL databases. The learning curve is not too difficult since it is mostly just uploading our site files onto the hosting website.

2. NAU's own server hosting:

   NAU has its own server hosting that it is paying for. Since one of our clients works at NAU, we may be able to request access from IT to some server space for our client's website and database. These server and database spaces would likely be free. There also wouldn't be many blocks to resources such as multiple databases, bandwidth, and storage. The database types would be limited to Oracle and Microsoft Access. The need to be link which requires using Cold Fusion. Framework development is also restricted to ASP.Net. All these features would require significant learning efforts from our team.

3. Heroku:

   Heroku is a cloud platform hosting service. Most of its services are paid for but it allows the creation of a small sandbox for web and application development. Heroku has a lot of database support for multiple versions of MySQL but the site RAM is limited to 512MB. Storage would need to be mostly done in a database since Heroku doesn't support much permanent storage. Bandwidth and performance metrics are managed by Heroku. While the site is most compatible with Ruby, it does support other languages such as Python and Java. The learning curve should not be difficult since it allows Git integration to upload files onto the website.

4.4.2 Chosen Approach

Our chosen approach for website hosting is Heroku. The table below outlines some of the reasons why we think Heroku is better than the other options. Freehosting.com seems to provide simple front-end hosting and features but is limited in mostly all other features. NAU's server hosting seems like the intuitive option, but would have a high learning curve due to us needing to learn how to use new frameworks and databases that our team is not experienced in. We would also need to manage connecting our pages to our database by learning how to use Cold Fusion. Heroku appears to be the best option because it has language support for languages our team is experienced in. It has enough features in the limited free version that would be enough to host and test a small website. Heroku allows integration and deployment from Git It has support for multiple types of databases as well.

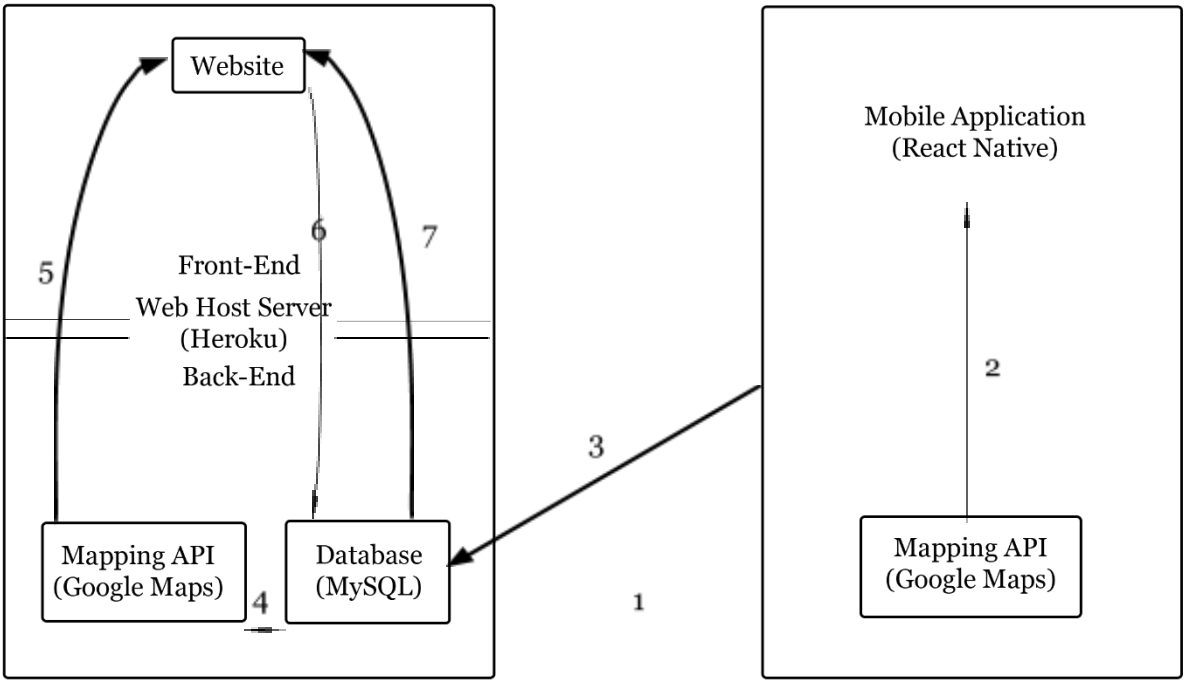
Web Hosting Services Comparison Table					
Scale 1-5 (5 is best)	**Pricing**	**Resource Availability**	**Learning Curve/ Familiarity**	**Database Support**	**Total**
Freehosting	3	3	4	4	**14**
NAU server hosting	5	5	1	3	**14**
Heroku	3	4	3	5	**15**

4.4.3 Proving Feasibility

Since the main issue of this challenge is website hosting with database support, our best way to prove the feasibility of our solution is to test a small website that we have created and then upload it to Heroku. Once we have set up our database, we can try to link it to the test website that we have uploaded. We can test this by adding basic database management functions and checking to see if these functions are actually interacting with our chosen database. It should be straightforward enough to create accounts on Heroku since it offers a free plan with basic features.

5. Technology Integration

Figure 1 is a basic representation of how we want to integrate our technologies. The website and application are front-end methods that users will use to interact with the database. The website will be implemented in standard HTML, CSS, JavaScript, PHP, etc., as there are not many alternatives to create a website. Our application will be programmed using the React Native framework and the database will be running on MySQL. The application will have to be able to upload statistics and data to the database as it is being used. It must also be able to read database data in real time as other users update it. The website will also update the database by registering users to the system. However, the website will mostly be reading map and user data to display statistics. Finally, both the website and application must be able to work with and display Google Maps. The website must be able to use Google Maps to display map statistics and data, while the application should be able to use the map to display live issues and problem areas that users can track around a college campus.



(**Figure 1**. Image representation of how our technologies will interact with each other.)

1. Data will be sent from the database to Google Maps (App). Number 4 nearly identical
2. Google Maps will use the data received to define regions and place markers.
3. Input from the application will be sent to the database and disseminated from there.
4. Google Maps (back-end) will be doing identical tasks as in the mobile application.
5. The website will be updated with the data received.
6. Website will send data to database to update regions and markers as needed.
7. The database should be able to send any other stored data to the website as needed.

6. Conclusion

We are developing a mobile application that will help encourage NAU students, staff, and faculty to actively engage in maintaining the campus. Cleaning litter in the campus, fixing broken fixtures, and clearing trash bins all help contribute to NAU's sustainability efforts. The features we are including will assist NAU staff by sending reports and tracking statistics based on user feedback.

In order to complete this task, we had to determine the best tools we could utilize when creating this application. This document has shown each of our solutions to meet the requirements for a minimum viable product. Our selected DBMS, MySQL, will be able to hold all the data-types needed and has proven itself reliable as seen by its immense popularity. MySQL can store and receive data to all the other software we have chosen. Our Mapping API, Google Maps, is well documented and has many functionalities which

will enable us to mark regions upon static maps, use on click functions, generate heatmaps from user data, and much more. With all the features included, the mass usage of Google Maps, and cross-platform ability, using it as our Mapping API is absolutely the right choice for our application. Building our application in React Native will have its challenges as we learn but we believe that developing in an environment that has cross-platform compatibility is worth the effort. Currently, our goal is to create our MVP for Android use only but since we are using this development platform, we leave open the possibility of pushing to iOS devices. Our web hosting service, Heroku, has all the features we could need for developing the back-end and front-end of this application.

Challenge	Proposed Solution	Confidence Level (out of 5)
Web Hosting	Heroku	4
Language	React Native	4
Database	MySQL	4
Map API	Google Maps	5

We believe that we can successfully create this mobile application with our chosen technologies. Each tool presented meets or exceeds our minimum requirements and allows for future development beyond our MVP.