GreenAZ: Software Design Document

Ryan Demboski, Justin Eggan, Jack Gilliam

CS-486C, Northern Arizona University

Instructor: Michael Leverington

Mentor: Vahid Fard

Sponsor: Richard Rushforth

Februrary 17, 2023

v1.0

Contents

Introduction

The state of Arizona is currently one of the lowest ranking of the fifty continental states for recycling waste and waste management. According to a study performed by *LawnStarter,* Arizona sits presently at the 47th worst state for recycling - with the Ball Corporation citing that the state has an 18 percent recycling rate. This problem only contributes to the massive problem in the country and world regarding handling waste and, more importantly, recycling. While some Arizona municipalities have been able to invest millions of dollars into infrastructure to avoid filling new landfills, smaller municipalities cannot meet this.

Our project sponsor, Dr. Rushforth, is an Assistant Research Professor at Northern Arizona University, and is working in tandem with the Arizona Board of Regents at ASU to deliver a complete software solution to help mitigate the current waste problem in our state. GreenAZ is to aid in improving Arizona waste management by using Arizona Recycling Potential data, and helping their team visualize that data in a more convenient manner than just viewing a cluttered, confusing excel spreadsheet. There needs to be a better process for viewing what is going on with waste management in our state. A process that is publicly accessible, has a clean and professional user interface, and engineered from the ground up to be simple to understand.

Since taking the reins on this project in August 2022, GreenAZ has thoroughly researched the pros and cons of available web technologies, and ultimately settled on a solution that can best be used to meet all the desired characteristics of the project set by Richard Rushforth and the Arizona Board of Regents. In this document, we will go into complete detail on what these technologies are, how they work, and how they will be used to reach our end goals.

Implementation Overview

Our web-based visualization system will use data gathered from the previous phases of our project. This data we will be using includes an inventory of all Arizona communities and their associated demographics, waste management and recycling service availability, and associated economic development metrics. The data also includes waste processing infrastructure and logistics of the policies, practices, and partnerships in place regarding recycling services in Arizona. Accompanying this data, we will also look at the data of end markets for recycling materials, as well as the economic impact of recycling at the national and state levels. This data is critical for our system to reliably create visualizations of the recycling trends in Arizona and be able to compare Arizona's current recycling economic performance with other states and regions. Our system will be able to use the existing data given to us and process it into data about what recycling materials are available or not available at each Arizona municipality. Using this processed data, our system will be able to create multiple types of easy to read graphs, such as pie charts, bar graphs, and line graphs.

Some key characteristics of our planned MVP implementation will include:

- Allowing users to interact with a map of Arizona municipalities.
- Making each municipality display data about what recyclable materials are available and not available in that location.
- Categorizing different types of available materials in the map for each location
- Using that data to display it via different types of charts and graphs depending on what the user chooses to display it as.
- *Stretch goal:* Giving a qualified user the ability to import their own data or change the data inside the GIS map.

Architecture Overview

Part of the process of GreenAZ's research thus far into the project was deciding on the perfect technologies to utilize, in order to meet all of the requirements set for us from the beginning. Dr. Rushforth made it clear in his proposal what we need to accomplish to be an acceptable minimum viable product. After much consideration, we've settled on a modern and relevant web stack that is increasingly popular in 2023.

GreenAZ's chosen web stack is commonly referred to in industry as the MEAN stack. In this method of developing full stack web applications, the acronym MEAN stands for MongoDB, Express.JS, AngularJS, and Node.JS. Each of these technologies does something completely different and helps form our application into a whole.

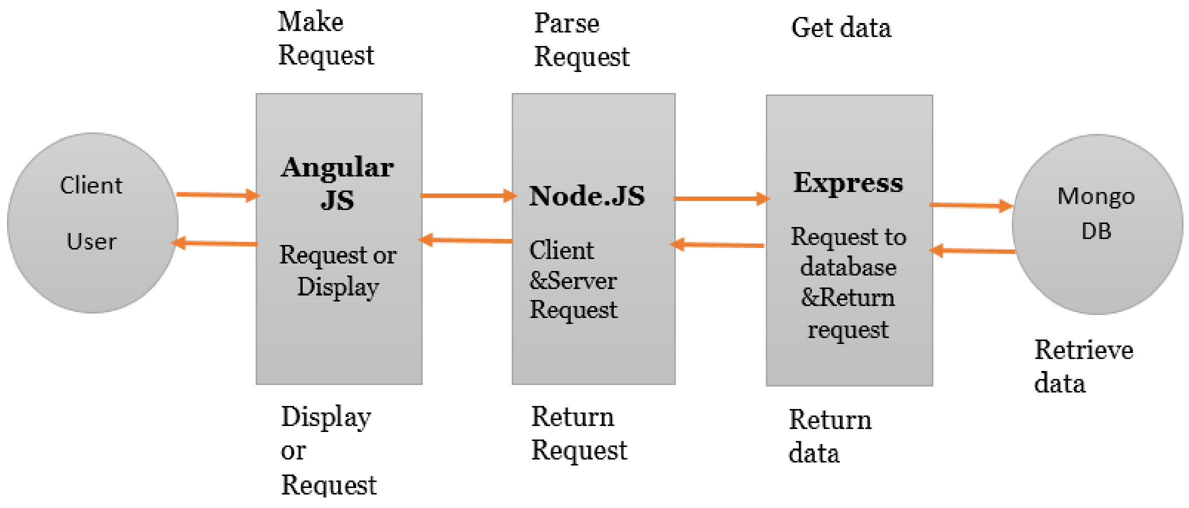


Fig. (1): MEAN Stack - Credit http://blog.skillbakery.com/2020/06/fundamentals-of-mean-stack.html

As shown in Figure 1, every element of the stack has its own distinct purpose. AngularJS, our front-end framework, is the top of the stack, and handles HTTP requests from the back-end to display to the client/user. It also allows the client/user to send HTTP requests from the front-end to the back-end. Lastly, it serves as a method of delivering customizable Javascript, HTML, and CSS code to the end-user's web browser, to create a great looking user interface.

Node.JS sits in the very middle of the stack, and it acts as a gateway to accessing both the front-end and the back-end of the application. In essence, Node allows developers to execute Javascript code outside of a traditional web browser. Without it, the stack would have no way to communicate with the HTTP protocol, thus having no way for our client to get any information on the website.

Express.JS is the second-to-last component of the stack. Express is needed to connect to our database system, as well as connecting to external APIs (Application Programming Interfaces). As an example, we can connect to an API that provides dynamic Arizona waste data, allowing our website to use datasets that update automatically every day, week, month, and more. Lastly, Express sends requests from the Node server to the database, and vice-versa.

Finally, MongoDB resides on the very bottom of the stack. This NoSQL database system allows GreenAZ to import real recycling data from the Arizona Board of Regents, and display it to our end-users in innovative ways on our website. Its the foundation of the project, and gives us the ability to load and store our data in a simple way that conforms to the rest of the stack.

With this MEAN stack approach, we are able to mainly use Javascript code, rather than multiple types of back-end languages in our source code, in order to keep the entire application consistent, maintainable, and well optimized. It's perfect for this project because it will eventually be given to another capstone team for further development. This architecture allows every future developer to get up to speed with the code quickly, as well as enable them to contribute to new requirements with nearly pure Javascript code in all phases of the application as a whole.

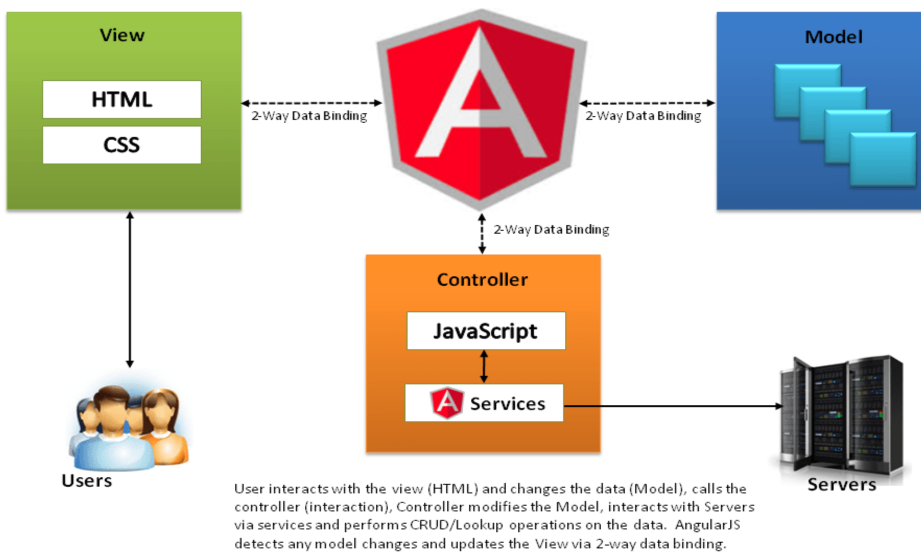
<u>**Module & Interface Descriptions**</u>

<u>Front-end Technologies</u>

We need a way to present our web application in a web browser. We need a tool that is able to communicate with the back-end in order to access our data and display it in our online web application. Since we are primarily using Javascript, we need a tool that can handle client-side Javascript code, along with HTML and CSS, to produce customizable web pages.

In order to compare the viability of each front-end web application development tool, we did extensive research on when each option works best. We have decided to choose Angular as the tool we will use when developing the front-end of our web application.

Angular is a Model-View-Controller framework that is used as a front-end interface to create dynamic web applications. Each web page is a standalone Angular project that can be hosted on an HTTP server, in our case we will be using Node.JS and its built in http-server package to host our Angular project.



User interacts with the view (HTML) and changes the data (Model), calls the controller (interaction), Controller modifies the Model, interacts with Servers via services and performs CRUD/Lookup operations on the data. AngularJS detects any model changes and updates the View via 2-way data binding.

Fig. (2): Angular Diagram - Credit https://avaldes.com/angularjs-introduction-and-sample-programs/

Shown above in Figure 2 is an example of how a typical Angular project works internally. The first component is the Model class, which is an object mapper class that translates the JSON packets from the database queries into fully javascript compatible objects on the front-end (see Back-end Technologies section for more details on this topic). It gives developers the ability to create database objects in their front-end code without needing to write complicated SQL (Structured Query Language) queries to the database. It makes the whole process much simpler for consumption of data on our front-end.

Next, Views are the second component of the Angular class architecture. These classes are the actual web pages that the end-user will see when they visit our website. These pages don't include any code that communicates with our back-end system, rather they are focused on delivering our user interface to the web browser – styled however we want to make them look. These files contain all of the HTML and CSS code, as well as client-side Javascript code, like having forms, buttons, or other interactive features that you may have seen on other websites.

The final component of an Angular project is the Controller class. Essentially, this is the brain of the web page. Writing server-side Javascript code in the controller allows us to come up with any functionality we want for our page. If we want a login/create account system, we write that functionality in the controller. This means the Controller is in charge of routing the user to different pages, communicating with our back-end system, and is how we add the functionality code for the different models and views we create, to make them do exactly what we intend them to do.

With these three features making up the full Model-View-Controller architecture that Angular provides, we now have everything we need to deliver the full-stack web experience that our sponsor expects this project to become. Angular is most easily integrated with the other technologies we will use, making it the most desirable choice for us. Because Angular is capable of being quick and dynamic, it will serve as a great way for our team to display our web application in a modern and professional way.

Back-end Technologies

Every web application that relies on displaying dynamic content and data to the user must have a back-end system. A back-end system connects a web server to a database system to dynamically add, delete, or view data on a database directly from the front-end interface of the web application. In our application specifically, the back-end will be responsible for importing recyclable material datasets into the database, to then be displayed on the website through various front-end technologies and techniques. Eventually, it should be able to allow the user to upload new datasets directly from the front-end into the database to ensure the user never has to interact directly with the actual database itself. The site should be fully user operable from the front-end alone. MongoDB, Express.JS, and Node.JS will help to achieve these requirements.

Starting off, MongoDB gives us a foundation to the entire application. Having obtained the data that was recently collected by the Arizona Board of Regents, we can import this data into MongoDB and setup schemas based on how we want the data to be stored (and how it can be queried by Express.JS). We can load and store this data in an efficient way and make it easily accessible throughout the entire application.

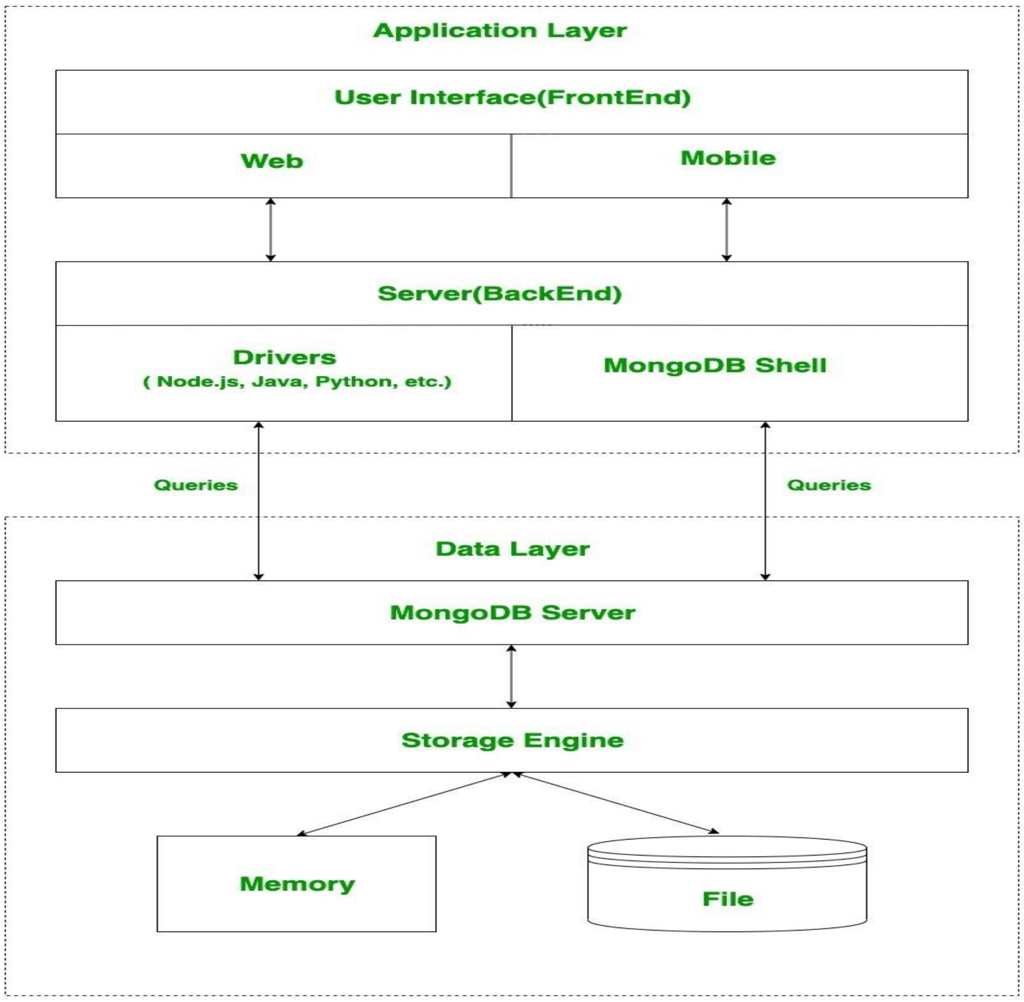
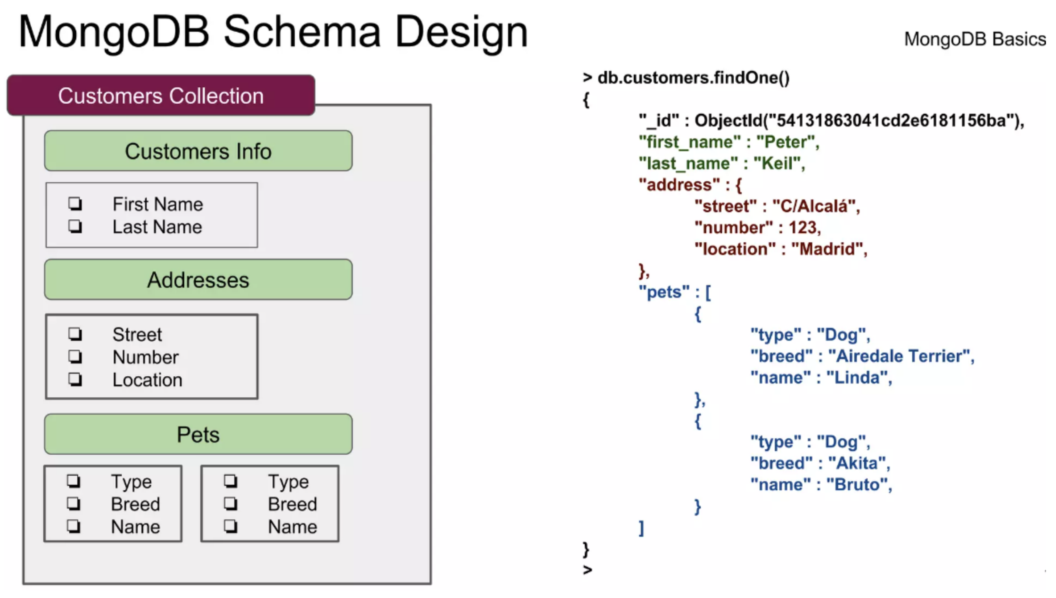


Fig. (3): MongoDB Diagram - Credit https://www.geeksforgeeks.org/how-mongodb-works/

Shown in Figure 3, MongoDB consists of an Application Layer, and a Data Layer. Starting from the bottom, the Data Layer contains the MongoDB server instance itself, as well as the storage engine (which is where the ARP data will be kept). When the web server that is hosting the MongoDB instance queries the Data Layer, Mongo will reply with a JSON response that contains the data needed by the query. An example of this is shown below, where the query is asking to find one random customer within the dataset.



```
> db.customers.findOne()
{
        "_id" : ObjectId("54131863041cd2e6181156ba"),
        "first_name" : "Peter",
        "last_name" : "Keil",
        "address" : {
                "street" : "C/Alcalá",
                "number" : 123,
                "location" : "Madrid",
        },
        "pets" : [
                {
                        "type" : "Dog",
                        "breed" : "Airedale Terrier",
                        "name" : "Linda",
                },
                {
                        "type" : "Dog",
                        "breed" : "Akita",
                        "name" : "Bruto",
                }
        ]
}
>
```

Fig. (4): MongoDB Schema Design - Credit https://www.slideshare.net/juanroycouto/mongodb-basics-unileon

The Application Layer of MongoDB delivers the queries from the Data Layer to the front-end of the application, for the end-user to see in their web browser. In our case, we can query for the recycling data on our database and display it on our website, and then style it however we see fit. However, as shown above in Figure 0, The data query first needs to access the back-end system before it can be displayed to the end-user. This means Node.JS acts as the driver that hosts our database instance on our server, and Express.JS acts as our method of connecting to the database and routing the queries to/from our front-end interface.

Since MongoDB uses NoSQL schema design, thus opting for JSON snippets rather than SQL statements, it means every aspect of our application can use strictly Javascript code to achieve our goal of delivering content from the database to the front-end. Express.JS does all the work of mapping the JSON data to front-end Angular objects for us, as long as we correctly implement the Javascript code to do so.

GIS Technologies

As our project is a visualization system that aims to generate a map with data on it, the most obvious hurdle is that of picking the correct GIS tool to use for our project. GIS (Geographic Information System) tools are applications that allow the user to create maps of areas while also utilizing some kind of visual editor or tool to allow the user to mark specific areas or add tags in the creation of said specific areas. These tools fit our product's description perfectly, and are some of the main tools we need to build the product.

The GIS tool we chose to use for our product is QGIS, due to its open-source nature allowing us to freely use the tool without needing to purchase an expensive license, and it meeting the requirements we needed in a GIS tool. These requirements were simply the ability to create a map that spans over county, state, and ideally country levels, a way to store data inside of specific areas on the map, and a basic zoom feature. These three features were all provided by QGIS, which is also well-documented enough that future users of our product will be able to search up interactions that we might not have thought of for future versions.

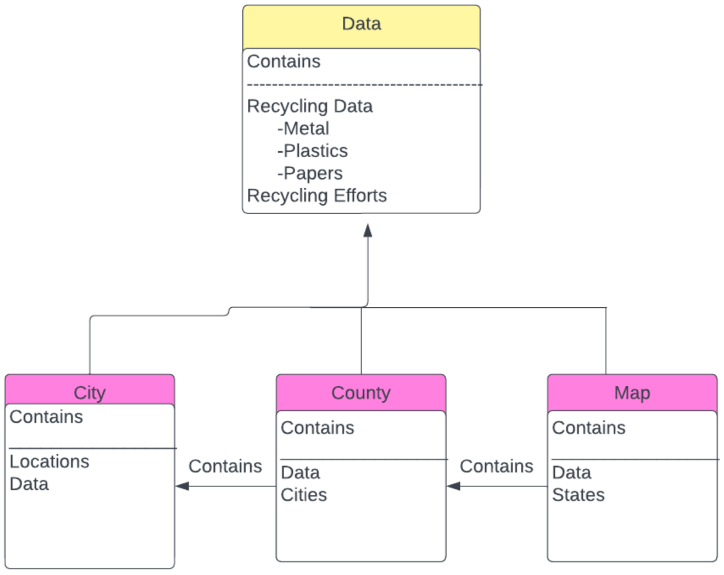


Fig. (5): A class Diagram of what data each part of the QGIS map entails.

As shown above in Figure 5, the QGIS systems are quite simple on how they build upon each other. The overall map contains the counties, each of which contain their respective cities within. Each level of the map contains the data for those areas, which includes recycling data and recycling efforts for those areas. The recycling data is a mix of how much paper is recycled, plastics, and other such efforts for recycling.

QGIS' public interface provides many tools necessary for our product. For one, it allows for integration into tools like Python and Javascript, which are the two main languages our product is built on. Next, it easily integrates with HTML and web pages which perfectly coincides with our final product's end goal. By having these features on it's public interface, it allows our product to become better overall.

Implementation Plan

The successful execution of our minimum viable product development will depend on a couple of key factors. Because we have already chosen the technologies needed for this project and have gotten familiar with each, we have now put together a concrete schedule that we will follow for development. Our front-end developer will build our web pages using Angular, HTML and CSS. Our back-end developer will create our database system, hook it up with Express.JS, and finally get the Node HTTP server up and running. Our GIS developer will utilize QGIS to prepare an interactive map of Arizona using our database, and export it as an HTML compatible element. Finally, all the pieces will be brought together to form a cohesive web application.

Once development starts, starting 2/20/23, some major milestones for us to look forward to during development will include:

1. Starting a Node web server that displays a “Hello World” Angular webpage.
2. Connecting the server to a MongoDB database and displaying dummy data from that database onto the Angular webpage.
3. Creating a test GIS map and a few graphs using that dummy data.
4. Importing the official data from the ARP team into our database.
5. Using that data to determine which municipalities we need to hardcode into the GIS map.
6. Creating a GIS map of Arizona using the municipalities from the ARP data and allowing users to interact with it to view the data that belongs to each location.
7. Implementing charts and graphs to display the data for each municipality.

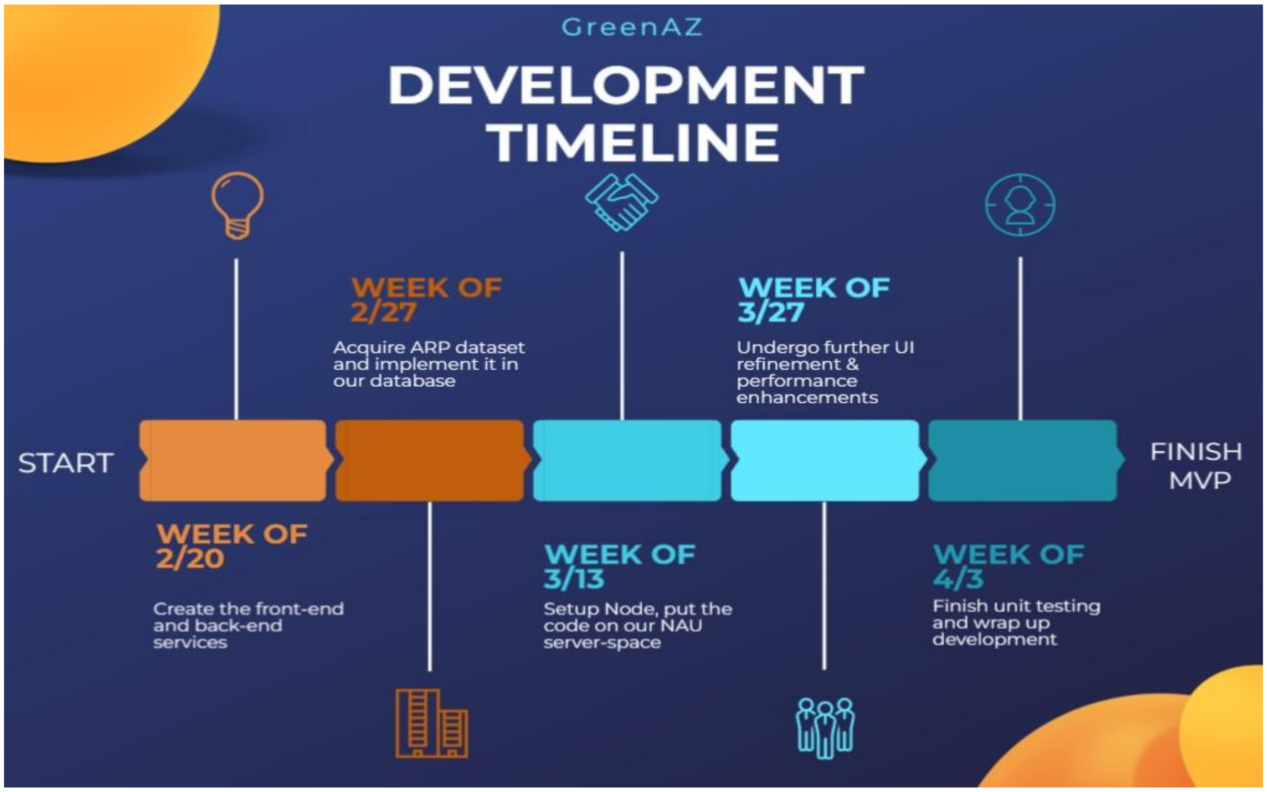

GreenAZ
DEVELOPMENT TIMELINE
START
WEEK OF 2/20
Create the front-end and back-end services
WEEK OF 2/27
Acquire ARP dataset and implement it in our database
WEEK OF 3/13
Setup Node, put the code on our NAU server-space
WEEK OF 3/27
Undergo further UI refinement & performance enhancements
WEEK OF 4/3
Finish unit testing and wrap up development
FINISH MVP

Fig. (6): GreenAZ Development Timeline

When these milestones have been completed, our project will have sufficiently met the minimum viable product requirements set by Dr Rushforth, and it will be delivered on time, before the designated due date.

<u>Conclusion</u>

In conclusion, the Arizona Recycling Potential (ARP) team has entrusted us with creating a visualization system to aid in the use of the Arizona Recycling Potential Model. The Arizona Recycling Potential Model seeks to help clean the state of Arizona due to it being one of the lowest ranked states for recycling waste. While some parts of the state have started trying to alleviate this issue, this visualization system will be a web-based system that displays a map that houses the requisite data and displays it through varying means. Our job, as a capstone team, is to produce this visualization system and allow the Arizona Recycling Potential team to input their data into the system for further use in their project with the Arizona Board of Regents in order to help clean up the state of Arizona.

So far, we have produced draft versions of each part we deem absolutely necessary for the final product. We have produced proof that our front-end, back-end, GIS technologies, and other visualization implementations can work. We are currently awaiting further data and server space from our sponsor, so that way we can start development using real data rather than using fictitious data. Once we receive the real data and server space from our sponsor, we plan on having development done as soon as possible, so that alpha and beta testing phases can start without further delay.

With our technologies chosen, and our documented plans on how we will code the project within this document, we are confident in making this product a reality. Our biggest goal aside from a successful minimum viable product is to make a product that is genuinely useful for real companies, counties, and communities outside of our university, to better understand Arizona's waste management needs and what they can do to help those needs.