CemetARy

Augmented Reality in the Cemetery

Senior Design Project

2019-04-17

Sponsor

Amy Giroux, U.S. Department of Veterans Affairs,

National Cemetery Administrator

Group 6

Jonathon Towne

Joshua Lecas

Katie Beason

Christian Monteverde

Table of Contents

1. Introduction

1.1 Narrative

This document contains the description, motivation, goals, objectives, and planning for the CemetARy project. Section 1 provides an introduction to the project, and a brief background on the need for the project. This section also provides insight into why the group decided to design a solution for the problem at hand.

1.2 Executive Summary

There are 136 national cemeteries in the United States, according to the Department of Veterans Affairs. Each of these cemeteries are home to the bodies of hundreds of veterans of numerous conflicts, and many of the biographies of these heroes are potentially extremely beneficial to the education of youth across the country. There are several issues with the access of information about these veterans.

Any visitor to a national cemetery can immediately recognize the issues involving finding the gravesites of the fallen heroes. The cemeteries are often extremely large, with several sections of the cemetery being home to non-veterans or other local deceased. One example of this is Orlando's own Greenwood Cemetery. The sections of the cemetery dedicated to veterans are not visible from the entrance, and the iconic headstones for veterans is easily lost amongst the sea of custom stones in the cemetery.

Once the visitor has found the section and gravesite they wish to learn about or visit, finding information about the veteran buried there can be a tedious process of Google searching and pouring over veteran biographies provided by the National Veterans Association. This can lead to visitors missing out on crucial information, and leads to boredom amongst groups of students. Our group has taken on the challenge to make the history of these soldiers easier to access, and to provide visitors to the cemeteries a user-friendly way to locate the veterans gravesites using their cell phones.

CemetARy is a mobile application dedicated to assisting and educating cemetery visitors across the country. There are two main features of the application. One feature is GPS and AR navigation to an indicated gravesite or cemetery section, and the other feature deals with delivering veteran information once the user arrives at the grave. The user will be able to select a veteran from a searchable list, and by means of AR and GPS 2D navigation, be given accurate directions to a gravesite. The AR portion of the navigation is meant to make the navigation easier, as in some cases where simple 2D navigation on a map becomes tricky when the landscape is not necessarily clear. The user will have the option between 2D and AR navigation.

The second main focus of CemetARy is an Optical Character Recognition (OCR) scanning feature. The current system that the National Cemetery Administration has explored is an image recognition app powered by HP Reveal™. The app currently works, however it is plagued by issues related to the way Reveal™ recognizes images. In order to work correctly, the Reveal™ system must have several images of each headstone in varying lighting conditions, and even with those images to compare to, the result is not always quick or accurate due to angles and other factors. CemetARy attempts to optimize this process using

OCR to detect lines of text on headstones, and use that information to conduct a search for that veteran. Since OCR relies on detecting alphabetical characters and numbers instead of using images, the system will not require training images, and the information on the headstone will be more easily utilized. Once the information has been scanned, the app will prompt the user to read the biography of the veteran, access more information, learn about the project, etc.

Combining these two features will allow for a much more pleasant cemetery experience for visitors and students alike. Locating a gravesite will be as simple as following an icon on a map, and accessing the biographies of veterans will be as intuitive as pointing a cell phone at a headstone.

2. Motivations and Project Objectives

2.1 Personal Motivations

Joshua Lecas

My interest in the project stems from a joy of working on software that involves going outside of the office to research and design solutions. The cemetery mobile app involves bringing a system into the 21st century, offering students and visitors a helpful and informational guide to the cemeteries across the country. On the technical side, this project will expose the group to a more complex aspect of mobile and web development, an area of computer science which I have had a fantastic experience. I am personally excited to be working on this project because of the nationwide implications of implementing a system like this. As our sponsor Amy Giroux detailed in our first meeting, this system may potentially be used by every national veterans cemetery in the United States,

making the demand for a quality product quite high. My hope for this project is that I will be able to learn new skills in the web development field, particularly the back-end, and I want to hone the skills I have already learned from completing earlier projects.

Katie Beason

I chose this project because I saw it as a practical way to help people. Both of my grandfathers were veterans and they have both passed away at this point, so I could understand the need for a way to efficiently find tombstones and I love the idea of having an app to learn more about the veterans who were buried there.

This project has the potential of being implemented in cemeteries across the country, so it would allow my group to make an impact all over the nation. I also wanted the experience a project like this could give me, it would allow me to

learn how to develop complex apps, as well as gaining experience in AR, robot vision techniques, and GPS navigation. I have some knowledge of these topics, but many I have not worked much in before so I am excited to learn how to implement them.

Jonathon Towne

I am excited for Senior Design. In this two semester prelude to graduation, I am eager to improve my team skills and sharpen my cooperative senses. It is my hope that in senior design I will garner a wealth of experience that I will carry forward into my career after I leave UCF.

Throughout my years at UCF I have read and learned a lot from book studies. But as everyone knows, there is a gulf of difference between knowledge and understanding. This project will demonstrate practical application of my education

at this university. In our project, AR and Navigation in the Cemetery, we have been tasked with creating an effective solution to our sponsors real world problem. This project presents an opportunity for me to leave a lasting effect on people in the world.

Unlike many projects proposed by sponsors this semester, this project will primarily be deployed outdoors. Of particular interest to me, our project will interact directly with the real world. Excitingly, the app we are going to be developing will be of tangible, material benefit to people navigating the real world.

Christian Monteverde

This project caught my interest because it involves the use of augmented reality and GPS navigation to help people navigate veteran cemeteries and view detailed information on the soldiers that have served and died in service of our country. I have a lot of love and respect for our soldiers and veterans, so given

the opportunity to help tell their stories while creating a great piece of technology is satisfying for me. These soldiers have sacrificed more than we can ever know for the freedoms that we have today. So this project not only gives me the ability to develop an amazing application that is useful to the world, but also gives me an opportunity to give a little something back to the veterans that have done so much for us.

Furthermore, this project gives me a chance to use the knowledge and skills that I have obtained throughout my time taking courses here at UCF and apply those skills to an important real-world problem that will be used by many people. I like the fact that this project is different from the other assignments that have been required of me from past courses. This project is not only made to receive a grade in a certain course. This is a project that will serve as a basis for other cemeteries to utilize and apply them to their own cemeteries that are all over the nation.

One of my main goals as a computer scientist is to create great software that is useful for people and makes people's lives just a little bit easier. This project gives me the opportunity to do that in several different ways. The fact that this project will be used by several people all over the nation and not just a small group of people was a big reason why I chose this project. I also have never worked with augmented reality or GPS navigation, both of which I find to be interesting pieces of technology and would love to learn how to utilize and apply to a project. I look forward to this challenge and the knowledge and skills I will obtain as a result of this Senior Design project.

2.2 Objective

The objective for this project is for the team to apply our collective skills in the field of computer science to solve the real-world problem presented to us by the sponsor. In addition to this, we will also create a professional, user-friendly tool which is not only easy to use, but also educational and engaging to users of all ages. We will create a lightweight application which is up to the modern day standards of web and mobile development, and provide the sponsor with the groundwork to easily scale the project out to multiple cemeteries with minimal effort required by the administration.

2.3 Goals

There were two main goals that we wanted to accomplish with this Senior Design project. The first was to create a user-friendly application that will help people navigate veteran cemeteries by leading them to the different gravesites of their choosing, as well as leading them to the specific individual headstones within those gravesites. The second, was to help educate users by using text recognition software to search our database full of stored information on each of the veterans and then displaying that information for users to view and learn from. The goal was to enhance the users experience when visiting the cemetery by making it easier to find the veterans and then providing them with the information to help tell the stories about these great American heroes.

The current cemetery that we used as a test case is difficult to navigate if you do not already know where everything is. The veteran sections are located all the way in the back of the cemetery, so people visiting currently have a difficult time finding those sections and searching the headstones to view the grave they would like to see. So, the goal was to help visitors find the different sections of the veteran cemeteries through GPS navigation and also help lead them to

specific headstones within those sections that they would like to learn about. Not only are there multiple sections containing veterans from several different eras and conflicts, but there are also hundreds of headstones in each of those sections. So, people visiting for the first time and looking for specific headstones that they would like to view and receive information on have a difficult time finding them. This application is meant to help people get directly to the headstones they want to see rather than them having to search through and read hundreds of headstones to find the right one.

Another goal of this project is to implement text recognition into the application that can recognize text that is engraved on the headstones. The cemetery currently has an application that uses image recognition on the headstones to figure out which soldier that the headstone belongs to and will use that to find the information stored in the database for that particular soldier. The problem with the current solution is that depending on the weather or the time of the day, the image recognition does not recognize the headstones properly.

For example, at times when there is poor lighting because of shadows or even when the sun is setting and gives off an orange light on the headstones, the image recognition software has a problem picking up and identifying the headstones and the information on it. This new solution is to implement text recognition in the application that does not rely on image color. This should solve the current issues they are having with lighting affecting the recognition of the headstones. The current goal is for our applications text recognition to work on the standard military headstones that are most commonly used today.

Furthermore, there will be a database that will store all of the detailed information on the soldiers as well as the GPS coordinates of the cemetery's sections and each of the individual headstones. There then needs to be an efficient search

algorithm that will receive the recognized text and search the database for the correct soldier so we are then able to access the information. Even in the case where there are missing characters in the names due to cracks or scratches on the headstones, the goal is that the search algorithm will still be able to find the correct information. The app will then display the information that is stored in the database for that particular soldier. This feature is included for educational purposes by helping users learn about these soldiers and the achievements and sacrifices that they have made for our country.

Finally, there will be a website that will strictly be used by authorized personnel only. The website will not be available for use to people that have not been previously granted permission by a current administrative user. This is of high importance because this website will be updating and manipulating the back-end database of the entire system. Unauthorized users should not have access to this type of functionality because that puts the database and the information stored in the database at risk. Each user signup will be vetted by a current administrative user and that user will need to approve that sign up before the new user has access to sign into the website. The functionality of the website is to enter in and maintain the veteran data as well as the GPS definitions that will be used for the navigation. The goal is to have secure website that also provides good functionality and is user friendly to all authorized users.

Some of the stretch goals of this project includes the text recognitions ability to not only recognize the current standard military headstones with horizontal text, but to also work with older non-standard headstones, such as headstones that use curved text instead of the standard horizontal text. In addition, another stretch goal is to include user navigation between each of the sections of the cemetery along the roadways that exist within the cemetery. Currently the roadways have not been mapped out by Google Maps so the goal would be to

map out those roadways ourselves someway to help visitors navigate the roads to find each section of the cemetery that they would like to visit.

In addition, the ability for the application to be usable offline is another stretch goal because not all of the cemeteries currently get cellular signals. Finally, the last stretch goal would be to completely enhance the users experience by providing a full tour of the cemetery, including taking query results from the users and creating the shortest path navigation to each of the sections and headstones in order and marking or removing marks from the already visited places so that the user does not accidentally revisit the same sections or headstones several times, as well as removing the clutter of marks from the users screen.

2.4 Broader Impacts

In the kickoff meeting with the sponsor, Dr. Giroux detailed the national impact that developing this application would have. Once the groundwork has been finished, the application will be pushed to the national level, and will eventually be implemented in every national veterans cemetery across the country. The app will provide the Administration with an effective and user-friendly way to digitize cemeteries without the requirement for thousands of photographs to be uploaded of each gravesite.

This will be incredibly useful for families of veterans, who may be extremely grateful to see their fallen family member’s story told to cemetery visitors. The impacts of the application reach into the classroom as well. Teachers and guest speakers will have the ability to bring photographs of different headstones into a classroom, and provide students with an interesting and fun way to learn about local heroes and the history of a range of conflicts. Students will be exposed to

multiple facets of Computer Science, with a focus on AR and Computer Vision. Exposure to these topics in a clean and useful fashion may inspire youth to explore a STEM field in the hopes of developing future enhancements on our nation's institutions.

3. Project Requirements

3.1 General Requirements

1. The app must be created in Unity for iOS and Android devices.
2. Accuracy in the navigation is important as many graves are close enough together that the GPS accuracy may encompass multiple graves and we are trying to get the visitor to the correct one without them having to wander around too much.
    a. Our sponsor has suggested ARCore andARKit are more accurate in terms of placing AR game objects at geographic coordinates but Vuforia may also be an option .
3. The database behind this app must be MySQL, the website front-end access to the database must have login security with a moderated sign up (meaning that you cannot just signup and get automatic access).
    a. The signup needs to be vetted and an admin user in the database would need to approve a new sign-up before the person can login.
    b. The database is to maintain the veteran data and GPS definitions necessary for the navigation.

4. The API to the database from the app must also be secure in some manner (possibly encrypted), but the end-user of the app should not be required to have a login.
5. Database will need to contain:
    a. Demographic information on each veteran in fields accessible to the app for searches (i.e. name, branch of service,date of birth/death, conflict, state,cemetery,section, Gravesite number).
    b. The sponsor shall provide an initial schema that the database will be designed around
    c. GPS information for individual headstones and section boundaries.

3.2 GPS Navigation Requirements

1. The App shall allow the user to limit the search by a selection of predefined filters.
2. Search filters shall allow filtering by name, cemetery, section, and conflict as a minimum.
3. The app shall allow the selection of AR vs 2D navigation to show them or guide them to the selected grave.

3.3 Text Recognition Requirements

1. The text recognition must give user feedback as it is working, whether that is outlining letters it is recognizing or some other "please wait" type of feature.
2. The text recognition cannot rely on color in the camera image for processing as outdoor light can vary greatly. The text recognition must

also be able to work on a photograph of a headstone, preferably grayscale.

3. The text recognition must work on standard military headstones.
4. Text recognition will search the database for matching item(s). Search algorithm needs to account for unrecognizable characters within the text and if multiple matches let the user choose.
5. If the app is running in a classroom, it should bring up all the choices that match the search criteria regardless of the cemetery they are in. Part of the database contents will be defining what a cemetery is, its boundaries, its sections, roads, etc.
6. Once a choice is made to display veteran information, the UI on the phone must allow user to access a web page and/or media that is available for the veteran.

4. Initial Planning

4.1 Initial Ideas

Joshua Lecas

My initial research into this project is focused on the database, so I have been concocting several ideas for what I will need to learn and research. The main issue I will need to tackle is how to represent a road in the database. Google Maps™ was not permitted to do accurate mapping inside of cemeteries, and as a result there are no named roads inside of them to navigate a person through. One idea is to try to use Google's API to return a JSON of directions, which we can display on the screen as a route. Another idea I will explore is saving different GPS points into the database at intersections of a road, and then conducting a BFS of the points to get the quickest path, with the final point being the point associated with the gravesite.

For the most part, I will follow the sponsor's schema in regards to the structure of the database regarding the veteran's info. MySQL will allow for simple searching based on the information obtained from the headstone scan. I will need to research which framework to host the database from, but I believe Flask may be the most appropriate option.

Katie Beason

The part of the project I will be focusing on is the GPS navigation to specific headstones. Initial ideas I have for this first include how it will be displayed in the app. We can do either a 2D, top down version of a map, or have an AR map

which displays through the user's camera. A 2D map could mark different sections in the cemetery by highlighting them different colors and labeling them. A line path could show the user the quickest direction to the graveside they wish to navigate to. The headstone, or headstones, they want to visit can be marked as stars or highlighted circles on the map. If the navigation is done through AR, there can be a compass at the bottom of the screen indicating which direction the user is heading in. It can turn red if the user is going in the wrong direction, and green if they are going in the right direction. Instead of a compass, there could also be an arrow pointing in the direction they wish to go. When they get close to the headstone they are navigating to, the approximate area of the headstone can be highlighted on the screen.

The functionality of the GPS navigation should reflect its design in the app. There should be GPS waypoints throughout the graveyard to make accurate navigation paths, and a GPS marker at each graveside in order to navigate to there. While GPS technology is not advanced enough to be able to lead the user to the exact position of the gravestone, it should lead them to a close range of it.

Jonathon Towne

One of our goals is to use live OCR algorithms to enable users to retrieve information about the deceased, such as a biography. One of my first ideas is to display a transparent rectangle over the live feed of the headstone alongside buttons below the feed relating to the database search parameters. This would allow the user to swipe over the text that they would like to search and by using one of the buttons to give our program context, quickly populate their search without having to fill out a form manually. This potential implementation would also prevent some conceivable user frustration at an inconsistent context detection algorithm. Many of the headstones in the cemetery have unusual

placements of text, some are even arched. Accounting for such unorthodox designs with an algorithm is a delicate problem that could be lessened to a great degree by employing user input. Relying on the user for some of the context parsing is potentially slower than an algorithm but may well be a more consistent, less frustrating user experience.

Another one of our goals is to use augmented reality to guide the user to their chosen gravesite with an app on their phone. There are many markers that we could possibly implement to signal to the user where the headstone they seek is kept. Our sponsored suggested a red arrow over the burial totem. My idea is to use a mobile marker that traverses the cemetery grounds alongside them, ensuring that the user is never lost. The marker itself could take on many forms, from whisp to friendly ghost, the digital guide would escort the user down the path, navigating the large cemetery grounds before coming to rest over the user's selected grave.

Once our users have sent their search request to the server, they may get back many more results than they need. Obviously, the idea result is one accurate search result. However, when we visited the cemetery that we will be using to test our project, many of the cemetery headstones do not have dates of birth or death, troop or squadron. In fact, I saw a few headstones that only had a single word on their face. Needless to say, some search results may not be immediately conclusive. In this case, it would be wise to enable the user to sort search results in more than simple alphabetical order. Search results should be sortable in alphabetical, birthdate, deathdate, service troop/squad, and location (both ascending and descending of course. This flexible sorting would have the added benefit of improving discoverability if someone wished to pay their respects to the rest of their troop/squadron.

Expanding on the idea of discoverability, another idea is to have certain keywords and phrases easily searchable from biography screen. Wherever in the biography there are dates or names displayed, we would allow the user to tap the highlighted text to immediately, cleanly fetch a search for relevant biographies. This would allow a user to easily find others that, for example, died on the same day as their beloved, were in the same troop, or even information relating to someone else mentioned in the biography.

Further exploration of “discoverability” brought me to the idea of using augmented reality to render the names of those buried in the cemetery in the air above their headstones. This would allow a user navigating the cemetery, perhaps being assisted by a virtual guide, to see the names of the deceased at a distance. I suspect that seeing the names clearly all around them may bestow the users with a greater sense of humanity and reverence. There are practical benefits as well, using this feature, users would be able to search for people that used to know in a more person way than a text based search, yet more quickly than wandering searching by eye.

As far as our OCR goal is concerned, running the OCR algorithm, whichever algorithm we decide on, on every frame of a video is almost certainly impractical or at the least efficient. Therefore, one of my ideas is to pause the live feed of the headstone to allow the algorithm to process as accurately as possible without disturbance. There are additional benefits to doing this. We could allow the user to use a contrast slider to emphasize the otherwise irregular features of the headstones. Many of the headstone in the cemetery have unusual fonts and inconsistent coloring.

Another way to improve the consistency and speed of our app could be to allow the user to select the relevant portions of the image they want the algorithm to

process. A quick crop/selection box could filter extraneous irrelevant data, such as shadows or trees that our OCR algorithm could potentially be mistake for letters or numbers.

Another facet of our project to consider is in-app navigation. Our project is an app with at least three main screens to navigate. However, there are many other potential screens that we may need to employ. Some of these could be: OCR, AR GPS, 2D GPS, manual search, about/instruction screen, general settings, search with a saved picture, previous search results, and saved/ favorited biographies. Because of all the potential screens we may use, I suggest that we use the scrollable popover menu design. AKA the established hamburger menu design. User should be able to recognize the iconic triple stacked lines and deduce the rest from there.

Christian Monteverde

Some of the ideas I have for the website portion of this project includes different ways to ensure a secure admin sign up. This website is intended to be used by vetted administrative users only, in order to update and maintain a database containing the information on the soldiers as well as the GPS coordinates of the gravesites and the individual headstones. Referring to the signup page itself, I plan to use HTML, CSS, and Bootstrap to create clean and professional sign in and sign up forms.

Our sponsor made it clear that the website must be secure. An idea that I have for security involves the case when a new user is attempting on signing up for an account. The idea is to send an email to an authorized user and notify them of the new user that is trying to obtain access to the website and the authorized user will then decide whether or not to grant the new user the ability to create an

account. A downside to this could be that the new user would have to wait for the authorized user to grant them access.

Another option would be to have some sort of an access code that is separate from the users account password that new users would need to enter at sign up in order to create a new account. I also plan on enabling CAPTCHA on sign up for added security to prevent any potential automated spam and abuse of the website. Touching on the subject of the web page that will be available to the user after a successful login, the user will have the different options of either entering the different GPS coordinates or the various information that pertains to each of the individual veterans. The format of the forms that the user will use to enter the relevant information will be set up in a way that agrees with the tables that are designed in the database to ensure that the new or updated information is stored in all of the proper rows and columns of the tables in the database.

My main initial idea for the website is to have a home page with a picture of the veteran cemetery and a description of the functionality of the website and the entire project as a whole. It will include the name of the project at the top and welcome users in a pleasant way. Then, I plan to have a navigation bar at the top of the pages to provide an easy way for users to navigate between the home page and the administrator sign in page. Then, when a user decides to navigate to the sign in page, they will be provided a page that includes a form for them to enter their login credentials including their account username and password.

In the event that a user does not currently have an account. I plan to have an option at the bottom of the form that will ask something along the lines of, “Don’t currently have an account?”, and after that question I will provide a link that the user can click on that will navigate them to a page that contains a form where they can create a new account.

This sign up page will include input boxes that will require the user to provide information about themselves that will be stored into the database back end of the system. The information that will be required of the new user includes attributes such as the users first name and last name, their personal email address, their desired username for the new account, as well as their desired password. This form will also potentially include the security measure that we have discussed with our sponsor as a possible solution which is an access code that the user will need to enter that should have been provided to them before creating their account.

Furthermore, I plan to ensure that the email that the user has provided actually is an active account that belongs to them. The main reason for this is that it is important that we are able to contact the user in case of any changes to the system or any new requirements regarding their account. This is to ensure that we are keeping them up to date on the current state of the system as a whole since this project will continue to be updated in the future. A way to ensure that this is a reliable email address that the user has access to is to have the user verify their email address. Something I'm sure most people are familiar with when it comes to creating a new account on any current system. The user will be sent a message to the email that they have provided which they will then have sign in to that email to retrieve that message. This message will include possibly something in the form of a link that the user can click that will navigate them to a page that will display text that will say something along the lines of, "Your email has been successfully verified.".

After the user has been navigated to that page through that email to our verification page, the user's email will be accepted and applied to their account information. Another option would be to temporarily send the user to another

form with an input box that requires a unique code the system generates that has been sent to the users email account. The user will need to provide this code and it must match the code that has been generated by the system. If the code ends up being a match, then the user will have successfully verified their email address.

Aside from the web pages that users will see before being signed into their accounts, we need to describe the rest of the functionality that is required of this website once the user has successfully logged into their account. From what I have gathered from what our sponsor has required of the website is that users will use this website to enter information into the backend database that is created for this system. The database will be used to store information on the veterans that are buried at these cemeteries. This includes information such as the veterans name, ranks, conflicts that they have fought in, and other interesting information on the veteran.

The database will also be storing GPS coordinates of the cemeteries including the coordinates of each of the gravesites within each cemetery as well as the coordinates of the individual headstones for each soldier that is located within these sections. This information will be utilized in the GPS navigation portion of the mobile application that is tied to the system.

The initial idea that I have for this is to provide the user with a form that contains several input boxes for each field that will be used to store this information into the database. The user will then have the freedom to enter any of the relevant information that they require whether it be updating currently stored information or adding completely new information in the case of a new cemetery being added into the system. They will be able to enter the information on the veterans as well as the GPS coordinates of their cemetery.

4.2 Block Diagram

The following is a diagram which details the separate functions of the project, as well as the group member assigned to research and design that section.

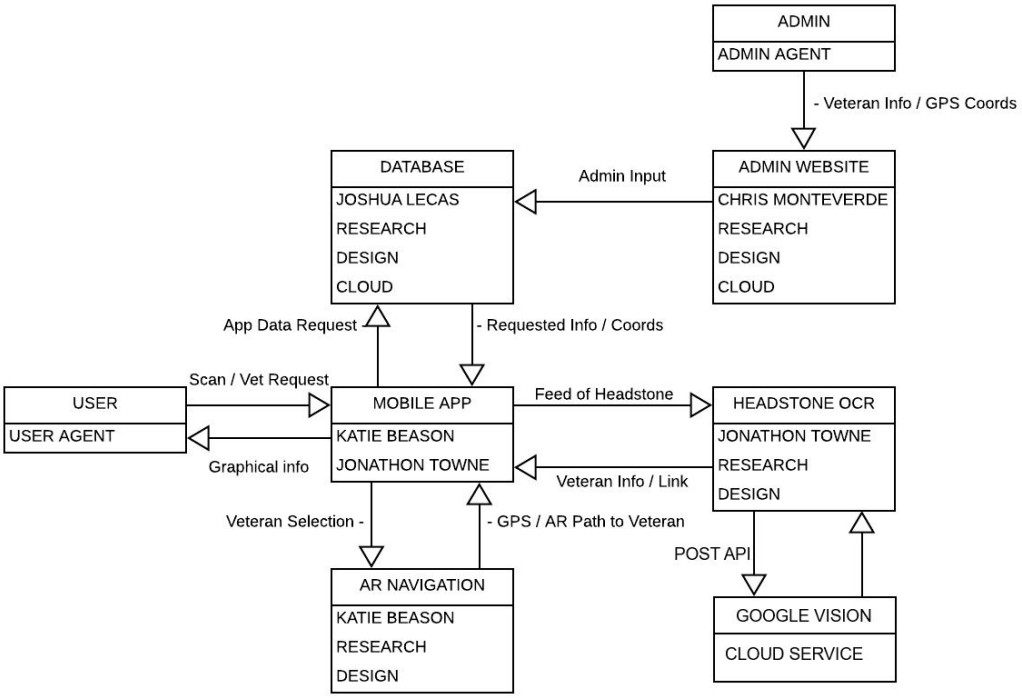

Figure 4.1 Block Diagram

4.3 Project Budget

The following table contains current costs in white, and potential future costs in gray. The current costs were all used in development for the application.

Asset	Cost
OpenCV	$95
In-App Web Browser	$12
Unity Storage	$5
Mapbox	$4/1000 monthly users (over first 25,000)
Google Vision	$1.50/month (after first 1000 units)
Heroku Hosting	$7/month (depending on the load on the database)
Play Store Availability	$25
Apple Store Availability	$99/year
CURRENT TOTAL	**$112**

4.4 Milestones

Below are the approximate milestones that will need to be completed over Senior Design I (Figure 4.2) and Senior Design II (Figure 4.3). The dates listed are dates agreed upon by the project sponsor and the team in Senior Design 1, and are an accurate timeline of the events which occurred in Senior Design 2.

4.4.1 Senior Design 1		
Date	Milestone	Status

2/3/2019	Initial meeting with Dr. Giroux (sponsor)	Completed
2/13/2019	Initial Design Document (15 Pages)	Completed
2/26/2019	Join ucfcs organization on GitHub	Completed
2/28/2019	TA Meeting 1 (8-10 Pages / Member)	Completed
3/1/2019	Develop initial website sign-in front end	Competed
3/4/2019	Heinrich Status Presentation and Meeting	Completed
3/22/2019	Design Document Individual Pages (15 per person)	Completed
4/10/2019	Submit Virtual Machine Request	Completed
4/15/2019	Construct Dummy Database	Completed
4/18/2019	Meet with Dr. Giroux for Database	Completed
4/20/2019	Connect server.js to dummy Database	Completed
4/20/2019	Construct Mockup UI	Completed
4/23/2019	Print Final Design Document	Completed
4/24/2019	Submit Final Design Document	Completed

Figure 4.2 Milestone Chart 1

4.4.2 Senior Design 2		
Date	Milestone	Status

8/24/2019	Meet with Heinrich	Completed
8/25/2019	OCR Camera	Completed
8/30/2019	Construct API Endpoints	Completed
9/10/2019	Complete API Testing	Completed
9/15/2019	OCR Character detection	Completed
9/20/2019	Fully Connect Website to Backend	Completed
9/25/2019	Fully Connect App to Backend	Completed
9/30/2019	Functional website completed	Completed
10/5/2019	OCR Character alignment	Completed
10/30/2019	OCR Character recognition	Completed
11/1/2019	Website Completed	Completed
11/1/2019	GPS Navigation implemented	Completed
11/5/2019	OCR Database Query	Completed
11/10/2019	Functional App Completed	Completed

11/20/2019	App Completed	Completed
11/30/2019	Testing Completed	Completed

Figure 4.3 Milestone Chart 2

6. Research

6.1 Web Design Research

6.1.1 Website Security

Our initial research into the design of the website is making sure that the website will be secure. We have already had some experience in web development, so developing the user interface should be pretty straight forward for us to implement. On the other hand, we have not had much experience when it comes to a website's security, so that aspect of the design will be the main focus for the website. Our sponsor made it clear that she wanted the website to be secure and that only authorized users will be able to create accounts and have access to the database update section of the website.

As a first step, when it comes to securing the website as a whole, it was found that using https is much more secure to use than just the regular http. With http, the browser will look up the IP address of the website that the user is trying to access and will connect to that address and it assumes that it is the correct one. After that, the data that is sent over the connection is presented as clear text that is easily readable.

There are two major problems with this. One is that there is no way to ensure that the address that the user has been connected to is the actual website that the user was trying to connect to. The address could be impersonating the actual website and this puts the user's personal information in danger. The impersonating website could ask for the user's login username and password and now the user has freely given the impersonator their login credentials without even knowing it.

The other problem is even if this is the correct website that the user was trying to access, the way that data is passed with http makes it easy for anyone eavesdropping on the network to intercept the flow of data that is being transferred back and forth. Since the data is presented as clear text, the eavesdropper will be able to read through the data and steal information without any issues. A good solution to this would be to use the https encryption that is available today. This is much more secure than regular http in that it will first check the websites security certificate and make sure that it was issued by a legitimate certificate agency. This helps the user feel safe on the website because there is a legitimate company that is vouching for the website that it is the correct one.

Another security feature that https provides is that it uses a code to encrypt the data that is being sent back and forth from user to server. The usefulness of this is that even if a hacker is able to eavesdrop on the network and get the information that is being sent, they would not be able to read or understand it because the information will be encrypted. We feel that this is essential to implement during the development process of the website.

The next security feature that the website must have pertains to new users that desire to create new accounts. The first option that we have discussed to ensure

that a new user has been approved authorization to make an account would be to first send an email to a current administrator when a new user attempts to create an account. The administrator would have to then either approve or reject the new user access to create an account. Although, the downside to this would be that a new user has to then wait for the response of the administrator and if the new user has work that needs to be done in a timely manner, they would not be able to access the website immediately, which could become an issue.

Another option that we have discussed that we think would be much more useful and efficient would be to have an access code that the user would need to enter in along with the other relevant information they would be entering in on the signup form. The idea is that the new authorized user would receive this access code beforehand from a current administrator of the entire system. This gives the new user the freedom and ability to create their personal accounts at a time that is convenient for them and enter the access code that is required that has already been provided to them, which will then give them instant access to the administrative features of the website and they can get to work on those features that they need as soon as they require them. The goal is to keep the website secure from users that have not been authorized access, but also to provide easy access to users that have been vetted and authorized to gain access.

Another area of security that we have researched and we think to be useful pertains to restrictions on user passwords. It was noticed that several websites, including the UCF website, require that a user's password be of a certain length as well as include special characters and numbers. We find this to be a useful feature for security because this makes it more difficult for hackers to guess user passwords. Hackers usually will initially try the most common used passwords that people tend to use such as a pet's name or a date of birth. If a hacker can obtain this information it makes it much easier for them to guess a user's

password. By adding a restriction on all user passwords that requires them to include special characters and numbers makes it much harder for hackers to guess a user's password even if they have personal information on the user.

6.1.2 Website User Interface

Aside from the much-needed security that the website must have, the user interface is another important aspect of the website. Our web developer has had some experience with developing web applications using HTML which will be useful for creating the web pages. It was also noticed that utilizing CSS style sheets are great to use to make the site look better and much more appealing. We do not want the website to be a simple white screen with text and text boxes on it. Using CSS will give the site a much better look and feel to catch the users eye.

Furthermore, the website will need to be able to communicate with the database that is created for the entire system. This is another important area of research because our web developer has not had much experience with connecting to and updating a database through a website that they have created. One programming language that has been researched to use in order to accomplish this task is PHP. PHP (or Hypertext Preprocessor) is a programming language that is widely used by web developers that have created, or are currently creating websites. Our web developer is more familiar with HTML so he mostly will be using that when designing the web pages. However, when it comes to connecting to the database, PHP is useful and is a great option to use to complete this task.

To add to the usefulness that we have found this language to be, this language can also be imbedded into the HTML source code that is created and can be

executed through it. In addition, our sponsor requested that the database required for this project be built using the MySQL database management system. This is another reason we are researching PHP because it provides the ability to connect to a MySQL database. It can create a connection to the server and check for any errors when attempting the connection. On a successful connection, PHP also provides the ability to create new tables in the database, insert values into those tables, and other great functionalities for updating a MySQL database. This language provides all of the database manipulation functionality that is required of this project.

6.2 Unity Research

According to the requirements, the app must be made in Unity, and be able to work with IOS and Android devices. Some of the members of the team have worked with Unity before, but research into specifically developing apps with Unity and working with AR with Unity was still required.

Unity is a useful game engine for developing even non-games (which is what our project will be doing). One of its great perks is that it can deploy to multiple platforms, which is necessary for our project so that it will be accessible to most people who go to the cemeteries. While developing natively for each platform can allow us to do more with the applications since each platform developer is made specifically for that platform, developing in Unity for multiple platforms saved us the time it would take to develop natively for each platform and allowed us to do more with AR, since Unity works well with AR already. Unity allows the user to incorporate 2D, 3D, media, and AR into the app, which were all needed components for our app. There are many AR tools already available in Unity for free, including ARCore, ARKit, and Vuforia, which is important since AR is a key

portion of our project. We will be discussing each of those AR tools further in the sections below.

Unity has a small learning curve, but since it has many tutorials and much documentation available it was relatively simple to use once we got started. It is also ideal if the sponsor wants to expand the scope of the project to more platforms in the future, because it is compatible with “Cross- BlackBerry 10, Windows Phone 8, Windows, OS X, Linux (mainly Ubuntu),Android, iOS, Unity Web Player (including Facebook), Adobe Flash, PlayStation 3, PlayStation 4, PlayStation Vita, Xbox 360, Xbox One, Wii U, and Wii”. It is also free to develop in Unity for these different platforms, which will be extremely beneficial for our project team and the sponsor as the app continues to be developed in the future. [1].

One issue with using Unity to develop the app is that Unity’s life cycle is synchronous. There is a free and open source software called eDriven that can allow Unity to be asynchronous is we need it to be, so this should not create an issue in the long run so we did not end up using it. It can also have some GUI problems with apps, but given Unity’s flexibility and overall usefulness with UI this also should be something we can work around. Unity can also cause long load times in certain apps and generally drains the user’s battery more than a natively-made app, but since our application will be focusing heavily in AR, this drawback was already anticipated. [1].

The sponsor has provided us with a Unity asset called AR+GPS. This asset should be sufficient for our augmented reality and GPS navigation needs, but it requires that we used either ARCore and ARKit, or Vuforia. Listed below are all of our narrowed down AR and GPS navigation options, including the AR+GPS Unity asset which we implemented in our project.

6.2.1 AR in Unity

A large reason our sponsor is having us use Unity to create the app is its integration with Augmented Reality (AR). In the app, AR is used to direct the user to specific geographic locations, which our sponsor believed that ARCore and ARKit were more accurate ways to place objects at geographic coordinates. This accuracy in navigation is vital to our project because we are trying to lead the user to specific headstones, which with the current GPS technology available can be a difficult task to get completely accurate. Because of this, it is important that we chose an AR developing tool for Unity which is more accurate with its geographic coordinates. Another noteworthy SDK for AR in Unity is Wikitude, which works better with GPS coordinates, but it is much more costly than the other options so we will not be focusing on it here. The newest versions of Unity now have ARCore, ARKit, and Vuforia built in, so it was a simple matter adding what we needed to the project.

ARCore

ARCore is a Software Development Kit (SDK) created by Google for Unity. It is made to develop AR functionality for Android applications. It specifically works for “Google Pixel or Pixel XL or Samsung Galaxy S8 running Android 7.0 Nougat and above”. ARCore does also work for IOS applications, running on the iPhone 6S or up and certain iPad devices, but it was created for Android use so it may not run as well on IOS. ARCore allows environmental understanding, which can allow us to place AR elements on real-life objects. It also allows motion tracking, which can let us place an AR element in a location in real-life and then walk around the virtual 3D object with AR (with it staying in the same real-world place on the screen). Both of these features were very useful for us in developing our application. [2, 3]

Another key capability that ARCore offers is light estimation, which allows the phone to estimate the environment's current lighting conditions. This feature was not needed for our project, but may be useful if the sponsor expands the project in the future. There are many tutorials and documentation available for this SDK, so it was easy to learn and start implementing. One great advantage of ARCore is that it is free to use, which will make development and future distribution much more cost-effective. [2, 3]

ARKit

ARKit, on the other hand, is an SDK primarily created for IOS devices. Unlike ARCore, ARKit is only compatible for IOS, so to make the AR work on Android devices we would also need to use ARKit in combination with ARCore. There is also a lot of documentation for ARKit available, so it was not a steep learning curve either. ARKit allows persistent AR experiences, which means that users can begin using AR and then come back to it at a later date, without having to completely restart the AR process all over again. It also allows shared AR experiences, and while that was not used in this project, it could be useful if the sponsor wishes to expand the project in the future. ARKit allows object detection and tracking, which was useful for detecting specific headstones in our application.

More features ARKit includes are "world tracking, pass-through camera rendering, horizontal and vertical plane detection and update, face tracking (requires iPhone X), image anchors, point cloud extraction, light estimation, and hit testing API", according to the ARKit plugin website. ARKit is free to use when beginning developing, but for distribution it costs $99 annually, which may be more than our sponsor wants to pay. Now that it is integrated in Unity though this cost may be waved. [4]

Vuforia

Vuforia is probably compatible with the most devices of the different AR software kits listed so far. It can be used on phones and tablets, including Android, IOS, and Windows 10, which includes all of the devices our application must be able to work on according to the requirements. It can also work with Smart Glasses and AR Headsets, which is not required for our project, but could be useful for later expansion of the project.

Vuforia allows marker-based tracking, which places AR objects on the phone screen based on a real-life object, such as a QR code. We would most likely be able to make use of the markerless tracking that Vuforia also supports, which places AR objects based on GPS location or position. From our research, GPS-based AR doesn't seem to be the intended purpose of Vuforia, but we have found some tutorials which are able to get GPS working with it. The basic version of Vuforia is free, which allows 1000 Targets and 1000/month Cloud Recognitions, but more advanced versions can cost either a $499 one-time fee, or $99 per month, which is much more expensive than the previous SDK's mentioned. [5]

Since ARKit and ARCore were better geared towards GPS we ended up deciding to go with those two options for our AR needs.

6.2.2 Unity Maps

A vital part of the Unity app integration was getting a maps SDK that works well in Unity. Since GPS navigation is one of the core requirements our sponsor needed, we needed to be able to have a GPS SDK that either created for us a

map or allowed us to add one in, and also allowed us to place GPS points on that map. The map must also update based on the user's current position, so that the user can track where they are in the cemetery. Different map SDK options that we found for Unity include GO Map and Mapbox. Unity also offers Online Maps from its asset store for $80, but we ended up choosing to focus on the other options to be more cost-efficient.

GO Maps

GO Maps is a map SDK that is compatible with Unity. It would allow us to create dynamic maps and location-based games or apps. It builds maps on mobile devices based around the user's GPS location. It also allows the user to set GPS points in the maps, both of which will be necessary features for the CemetARy app. It uses a number of other vector map APIs, including Mapbox, OpenStreetMap, esri, and Mapzen. It does cost $89.99 though, which is more than we ideally wanted to spend.

Mapbox

Mapbox, on the other hand, is a much more affordable SDK to use in Unity. The initial account with Mapbox is free, and allows up to 50,000 monthly active users, 50,000 Geocoding requests / month, 50,000 Directions requests / month, 50,000 Matrix elements / month, and 50,000 Tilequery requests / month for the mobile SDK. After that it is only $.050 per 500 monthly active users, 1,000 Geocoding requests, 1,000 Directions requests, 1,000 Matrix elements / month, and 1,000 Tilequery requests / month. Depending on the needs of the cemeteries, the free version may be able to support the demand for the app, but if they need to pay the additional fees for the greater amount of features then it should be affordable to do so. The pay-as-you-go plan for Mapbox includes in it Satellite & street maps, Mapbox Studio, 50 GB tileset storage, 5 GB dataset storage, Unlimited Studio styles, and Public/free web & mobile apps. Mapbox also allows

multi-platform support for Windows, Linux, MacOS, iOS, and Android, which covers both of the major platforms that we are required to support for our app.

Mapbox includes in it both Mapbox Streets and satellite imagery, which both allow the user to track their GPS location on a map. It also allows for custom datasets to be inputted in Mapbox, which are integrated into the maps. Mapbox was created to have high performance, so this should help account for the performance issues that having an AR map may cause and create a less frustrating experience for the user. Mapbox was also created for scalability, which will help ensure that the sponsor is able to scale the map from just the Greenwood cemetery, where we tested our project, to many more international cemeteries.

AR is also supported with Mapbox. It allows live location data to be transmitted to the app and lets points of interest be tagged. Many of the tutorials we found using ARKit, ARCore, or Vuforia also used Mapbox in order to track the GPS location of AR objects.

We ended up choosing to use Mapbox to develop our 2D Navigation maps, but we did not use it’s AR capabilities since the AR+GPS Library had more functionality in that area.

6.2.3 AR+GPS Library

The library that we want to focus on the most is the AR+GPS library provided to us by our sponsor. This package allows us to position 3D objects in real-world locations using their GPS coordinates. It allows the use of AR Foundation, Vuforia, ARKit and ARCore.

AR+GPS currently is set up to work with IOS and Android, which covers the phone platforms that we will be required to release on. This package is unfortunately still in its early versions, so it potentially could have had many bugs we needed to work around. That is the primary reason we included all of the other research into other SDKs and assets we could potentially use in case the AR+GPS package didn't work correctly for us.

According to the documentation that came with the asset, some of the main features of the package include:

- Placing 3D Objects in geographical positions defined by their latitude, longitude, and altitude
- Placing 3D Text markers on geographics points. This feature includes an example which uses the OpenStreetmaps asset
- Device location and heading updates
- Double precision vector structs

The package also comes with many sample scenes which show examples of how to position objects, place 3D points of interest on a map, make objects go along predefined routes on a map, and render an ARLocationPath. These scenes were useful in creating our own Unity scenes and assets with this package. An example of the UI in one of these scenes is shown below in Figure 6.1. [6]

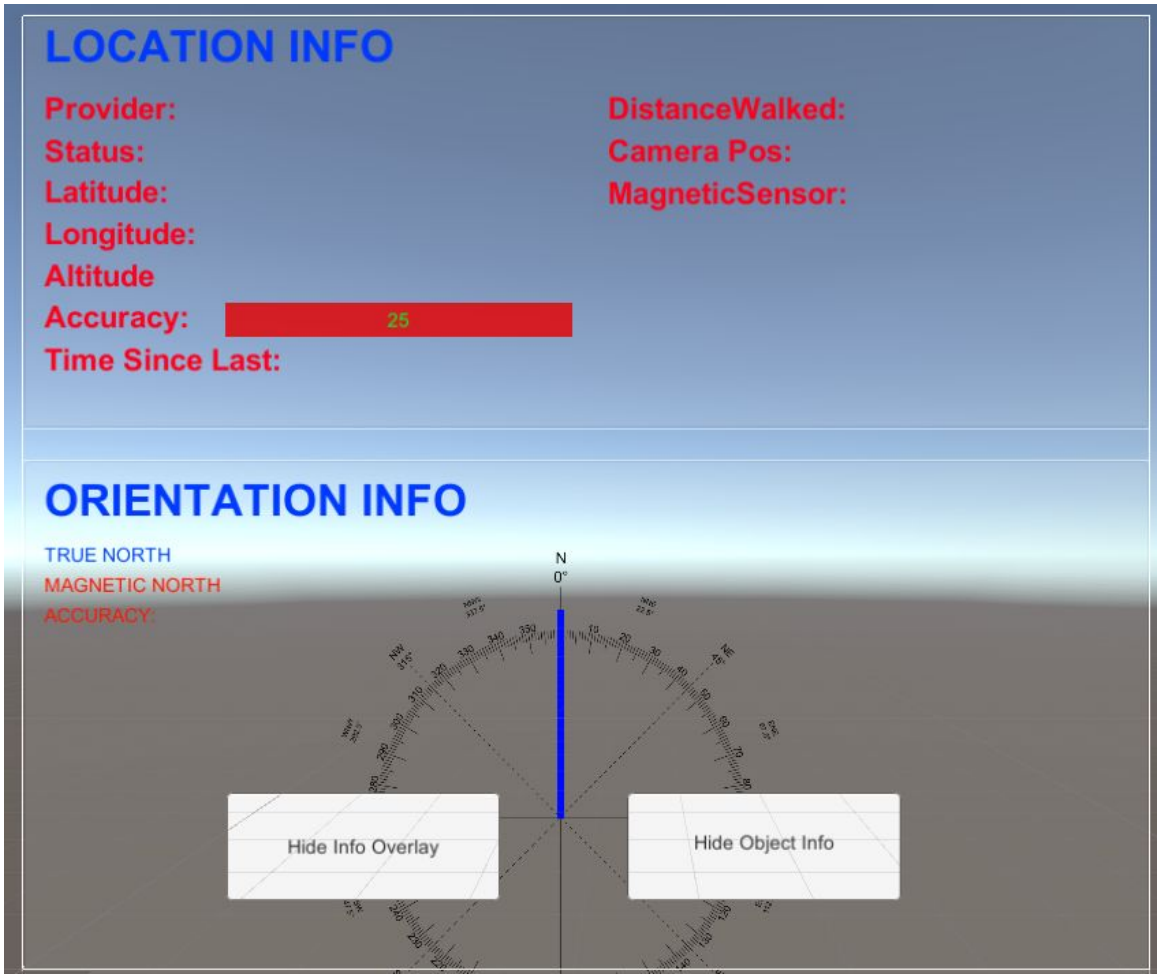

Figure 6.1 Sample of the UI from the AR+GPS package in the scene view.

This is how the UI appears in the Unity scene. While the information provided in this scene was not displayed in our final product, the information was stilled used behind the scenes in development. Figure 6.2 below shows what the sample scene looks like when played in the Unity editor.

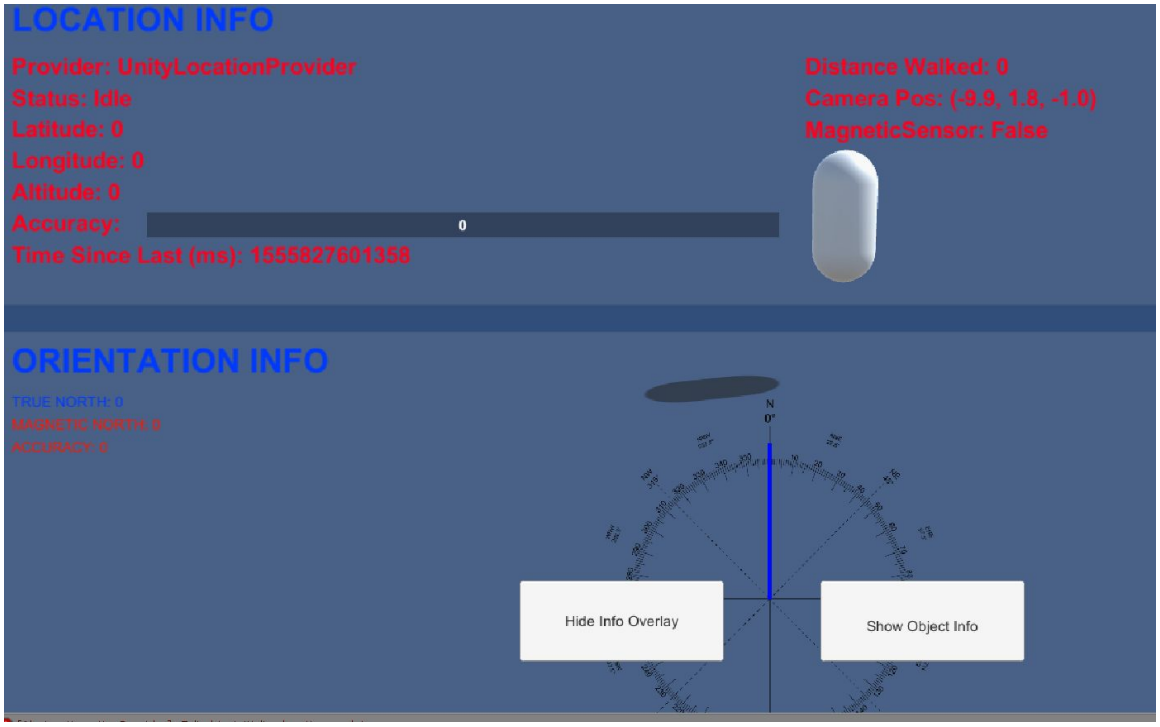

Figure 6.2 Sample of the UI and GPS tracking object from the AR+GPS package in the game view.

The image above shows a clearer representation of what the user will see when using the AR+GPS package. The capsule located in the scene is placed at a strict GPS position according to its latitude and longitude. In the game view in the Unity Engine we are able to walk around the capsule and it will stay in its position, and the same will occur with AR on a mobile device. AR on the mobile device will use the phone's camera to show the location of the object, and will update the location and orientation information.

The developer also included a Class Documentation section in his documentation of the package. This includes a list of most of the classes and details on their purpose and how to implement them. This was useful when we were developing the application and attempting to figure out what each of the scripts and classes do in the project.

The documentation for the AR+GPS package describes how to set up a basic scene with AR. It also gives summaries of the most important components and prefabs that come in the AR+GPS package. An important component used in the above scene example was the “ARLocationPlaceAtLocation” component, as shown in Figure 6.3 below.

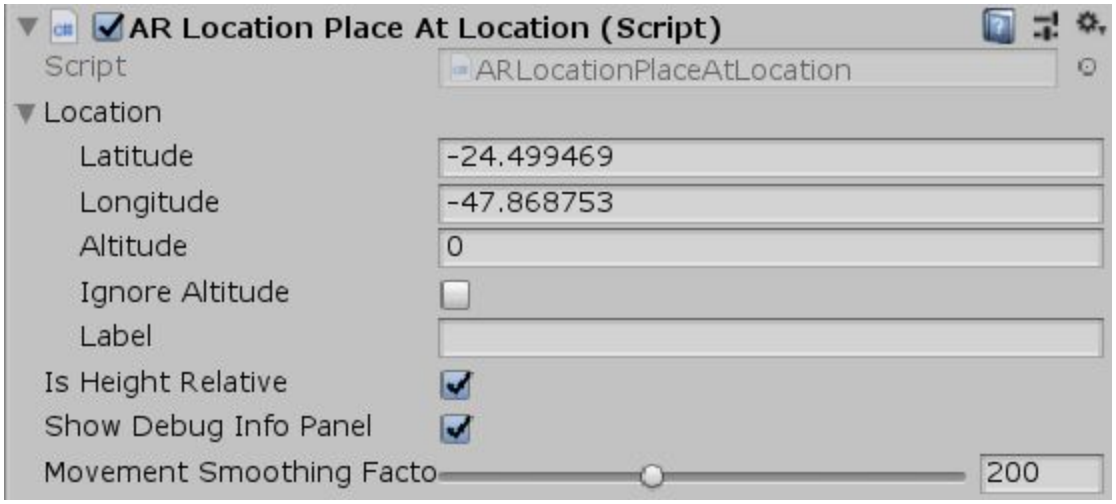

Figure 6.3 Example of the ARLocationPlaceAtLocation component in the Unity editor

This component was useful for us in developing our application because it stores the location latitude and longitude for objects that we needed for the AR portion of our application. The latitude and longitude of the location must be updated to match the exact geographic coordinates of any object we may want to place near a real-world object, or else it will not appear in the correct location.

The AR+GPS package has several limitations which the documentation in the package lists. These include:

- Imprecise altitude information
- Deterioration in the quality of scene orientation and true north direction after the user moves too far of a distance. There is currently as bypass for

this which involves resetting the AR Session after the user walks a certain distance from their initial position

- The object can jump around in the scene due to GPS precision causing the position data to jump around
- Movement smoothing has to be used lightly

All of these limitations can cause potential issues for our application since the application will be using most of these features. After reading the documentation it sounds like there are workarounds for many of these bugs, but they are by no means perfect. [6]

The developer of AR+GPS also plans to implement many more features of the package in future releases. These include ARKit plugin support, AR Hotspots, dynamic floor level calculation, and improving movement smoothing, among other features. To our knowledge our sponsor has been in contact with the developer of this package before, so we should be able to contact him to keep up to date on all of his future releases that we or the sponsor may be able to implement into the application. [7]

6.2.4 Unity API's

In order to make our Unity app effective and be able to work with the sponsor's needs, it must be able to connect to the database that we will have set up in order to store the veteran's information and the GPS points. API calls can be difficult in Unity, though, so this subject required research from us to ensure that we could access our database effectively from the app.

The database that we will be using is MySQL, so Unity needed to be able to make API calls and connect to this specific type of database. Research into this topic has indicated that this will be possible, even if there are sometimes

connection issues with MySQL, especially when initially attempting to set it up. Using PHP is a common suggestion for trying to do this. The Unity app in this case would be the “client”. The client itself should not directly access the database, because this can create security concerns. The client (so the app) instead should contact the server scripts, for instance, in PHP or ASP, which would then access the database. So our Unity app will only ever be contacting the PHP script, which will then access the database for it.

So far we have been able to find many tutorials online for connecting to databases in Unity, most of which are specifically geared towards MySQL. Most games and apps available these days require database access to be able to efficiently store player or game information, so it shouldn’t be hard for us to find out exactly how to code our Unity app to make it access the database.

We ended up converting our Unity code to JSON and sending that to the API, and converted the JSON code back into string format when information was received from the API. Unity had built in functions to convert from JSON to a string, so doing this wasn’t overly complicated.

6.3 Database Research

All database research, design, prototyping, execution, and deployment is assigned to Mr. Lecas. Section 5 explains the details required for the database as explained by the sponsor, as well as any initial planning, diagrams, or research into implementation. Subsequent chapters will detail the in-depth research and implementation that are chosen for the final design.

6.3.1 MySQL Requirement

As indicated in section 3.1, Dr. Giroux has requested that the team use MySQL as the backend database for the project. This allows the group to easily hand the project off after completion to the sponsor and any subsequent customers without the restriction of a lengthy explanation of differing database systems as a requirement. Dr. Giroux has extensive experience with MySQL, therefore it will allow for a smooth transition from working prototype to final production version. Having Dr. Giroux as a sponsor also allows for occasional insights and help with the database set up due to her knowledge of MySQL as a platform.

Dr. Giroux also has indicated that a previous schema for MySQL exists for the Veterans Legacy Program, and has provided the schema to the group as a guideline for the prototype database. As it stands, there are several tables included with the schema, to allow for extremely accurate data entry for each veteran.

6.3.2 Stored Data

The data we will be storing should be managed all under a single MySQL database. This includes the data for the website, the information regarding the veterans buried in the cemeteries, the cemeteries themselves, and the sections inside each cemetery associated with the buried veterans. Each table of data within the cemetery will contain several GPS location points. This will allow for the app to provide the necessary navigation by retrieving the data for a selected veteran, and the section of the cemetery they are buried in. At this time there is no plan to gather or store any data on the application users. Section 3.1 details that the application itself should not require a login for the user, and it should be ready to perform upon download. The database will also contain information on

the administrators, which will be simple account information: user name, password, etc.

6.3.3 Website Access

The website for the application deals strictly with administrative use only, and creating an account to access the website will be vetted and provided by the administration. In the Requirements for the project in section 3.1 explains that only an administrator registered in the database should be able to approve and provide accounts to newly registered users. There are hundreds of examples on the internet of website requesting email verification from their users, and blocks certain content until the email has been verified. This verification process usually involves clicking a link, answering a security question, or an access code the user will enter from their profile screen. Upon discussing the method of access control with the sponsor, the group decided that an access code would be the most efficient and user friendly way to ensure a moderated registration feature.

There are multiple ways to achieve this validation from the front-end perspective. The user could create an account first, and require a code in order to access any of the site features. The user could also require a code from the point of account registration as a required field. How the code is acquired by the administrator is also a design choice on the front end, however the method discussed at the second group meeting was an email sent to the administrator including a code generated by the back end. Details of the research and design of the website access will be discussed in a subsequent section. Section 5.4 deals with the back end solution ideas for website access.

From a back end perspective, an access key authentication system is fairly straightforward. The database can either pre-generate codes and store them in a

table, or generate a code upon request and then save them to a table. The latter option allows for easier maintenance, as the codes never need to be ‘regenerated’ if the administration grows. The keys should be of a standard length, however as the keys will be generated on an as-needed basis, there is no chance of key spoofing as long as the codes are marked as used in the database when they are called in the registration function.

The method of key access as discussed in our second meeting makes use of an email system to provide the administrator with a key to give to the registrant. For example, the user selects a “Request Access” button of some form, and the API endpoint connected to the button would generate a key, store it into a table, email the key as a string to the administrator, who can then provide it to the registrant in whichever communication method they deem necessary. Once the key is consumed in the registration process, a boolean value in the table is set to ensure no key can be used multiple times. Figure 6.4 on the following page explains this process visually.

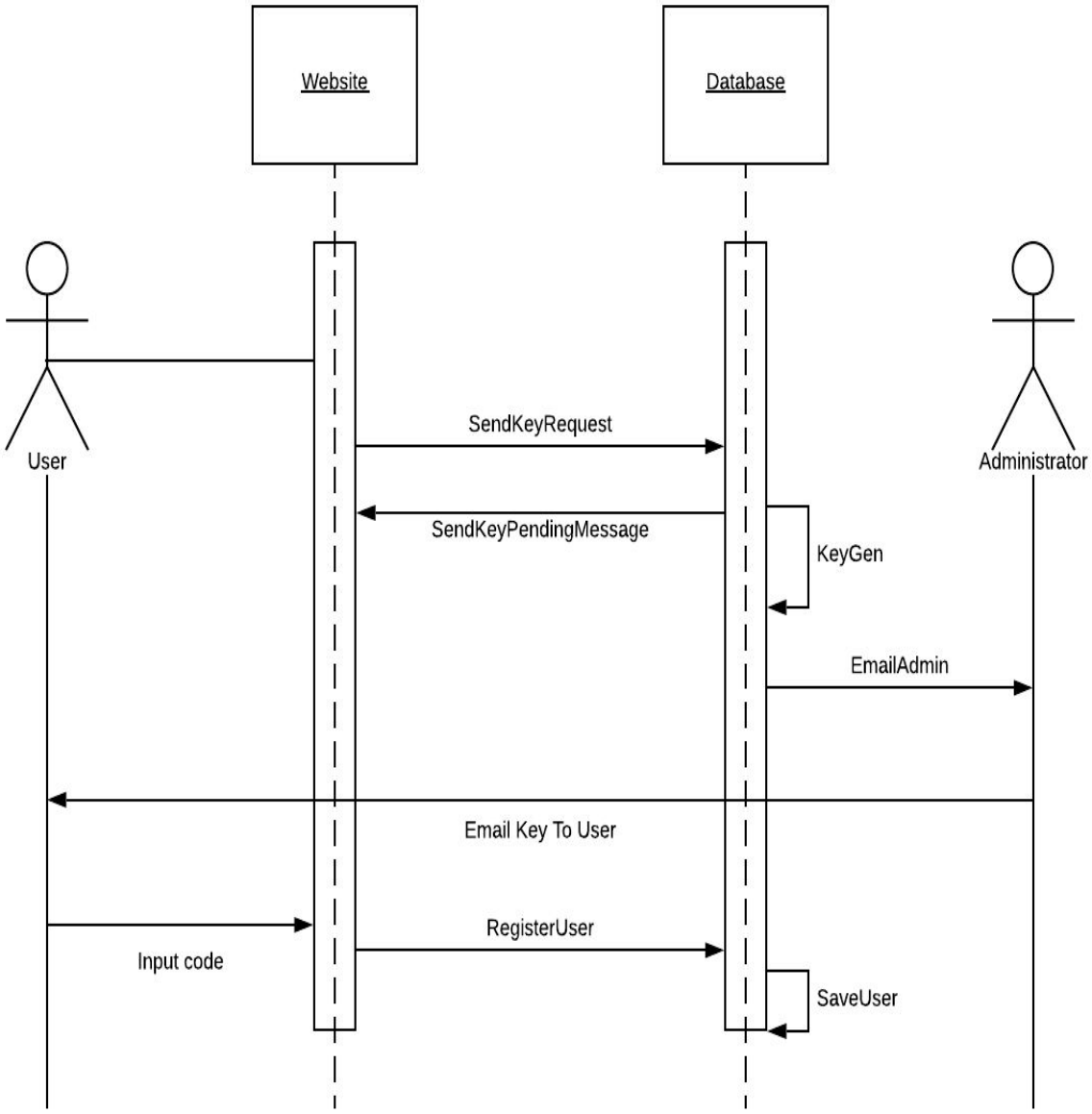

Figure 6.4 UML of Access Key Distribution

6.3.4 OCR Information Store

The database will also contain all of the information about the veterans buried in the cemetery. This data will be what is accessed whenever the user scans a headstone with the OCR feature of CemetARy. Each headstone contains limited

information about each veteran, for example: the full name, birth and death dates, the conflict fought in, their home state, etc. The database will have to account for this, and have an API endpoint to search based on the fields on the headstone. However, not every headstone contains the same information, and some headstones are damaged to the point where the information is not visible. The database will have to allow for this information to be omitted.

Figure 6.5 is an example of how the information on the headstone will connect to the different entries in the database. Headstone photos are by Dr. Giroux and used with permission.

Figure 6.5 Example OCR scan

The numbered boxes associate to the database as such:

1. **EMBLEM** - The religious emblem associated with the veteran.
2. **FIRST_NAME** - The first name of the veteran.
3. **MIDDLE_NAME** - The middle name of the veteran .
4. **LAST_NAME** - The last name of the veteran.
5. **STATE** - The veteran's state of enlistment.
6. **RANK** - The rank of the veteran.
7. **UNIT** - The unit in which the veteran served.
8. **CONFLICT** - The conflict in which the veteran fought.
9. **BIRTH_DATE** - Veteran's birth date.
10. **DEATH_DATE** - Veteran's death date.

Not every headstone contains the same amount of information, and in some cases such as the religious emblem, the information is known but not included on the headstone. The way to view the headstone lines is essentially to treat them as filters in a web search, the more information the OCR picks up, the easier it is to narrow down to a single veteran to display. Other fields of information, such as the section of the cemetery, site, row, etc, can all be obtained using either user input or GPS data from the phone itself. To be able to handle the information which is not included, we must use the SQL property **NOT NULL**, which allows for the information to not be required in a valid table.

The schema provided by the sponsor includes all possible known information about the veteran, and this data will be used by a search algorithm API endpoint to bring up the biography about the veteran. The schema also includes fields in the tables for information which is not included in our testing grounds, and these fields are to be used at the national level. CemetARy as a prototype will include these fields, but will not utilize them.

6.3.5 Navigation Data

The navigation portion of the project is where the group does not have a predetermined schema provided by the sponsor. The main goal of the navigation feature is to utilize the user's phone location and guide them to a specific headstone that they select via a search. The team discussed a few different ideas as far as navigation is concerned.

The base requirement is that a user is given a straight line direction from their phone to the headstone, by the use of an AR guide or a 2D path overlaid on a map. This can be achieved using only two points, the GPS location from the phone and the latitude/longitude point associated with the headstone. A stretch goal however, was to utilize a collection of points collected on the roads within the cemetery to provide a more accurate path, and simple navigation between sections.

The database will have a table for each cemetery, as well as a table of sections associated within that cemetery. These sections are often not perfectly rectangular, with some sections being irregular and not easily defined shapes. These sections can be defined by a collection of GPS points around the perimeter provided to us by the Cemetery Administration. One idea to accomplish a more accurate path to navigate between sections was to save the section data points, as well as the collection of points inside of the cemetery along the roads. Using a GPS point gathering device, one could create a list of points on the road, going by a standard of point collection. An example of this would be to collect longitude and latitude points at every intersection and every 45 degree and 90 degree angle turn. Once the points are collected, a navigation algorithm would utilize a common computer science node exploration method such as Breadth First Search or Depth First Search, and the application would then overlay a path along the GPS points dictated by the algorithm.

As stated previously, the navigation idea using a collection of points was a stretch goal, but was not completed in time for the writing of this document.

6.3.6 Hosting the Database

UCF has offered to host the database, as well as the web server, on a virtual machine located on the main campus. While there are other cloud hosting options available, such as AWS, Heroku, Google Cloud, and many others, Dr. Giroux has indicated that the UCF Virtual Machine solution would be suitable for the purposes of this project.

The only drawback to using the UCF network is that the network is private. In order to test and build the web server, we will need to connect to the UCF network via a VPN. Luckily, UCF offers a free VPN for all students, which is easy to download and use. The VPN is accessed by this URL: https://ucfvpn-1.vpn.ucf.edu/. Which then brings up the dialog box in Figure 6.6 below which allows for UCF students and faculty to log in.

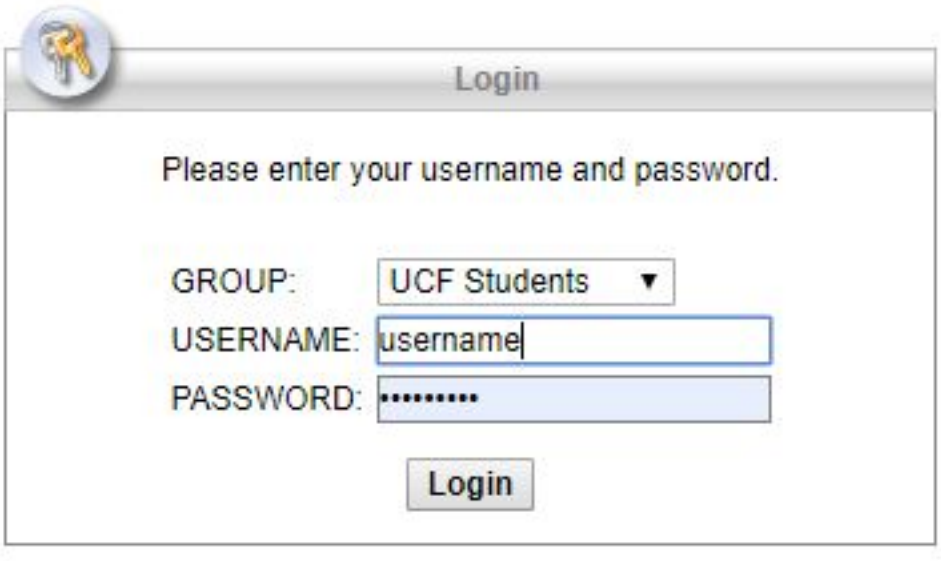

Figure 6.6 Logging into the VPN

In order to use the VPN, the user is then prompted to download and install the Cisco Anyconnect Client™.

In the final version of the application, the VM at UCF ended up not being functional enough to provide a successful project, and thus Heroku was used to host the database and webserver. Hosting on heroku was an effective way for us to deploy the project, as the web-server and website were located inside of a github repository, and whenever the project was pushed to master, Heroku would automatically build and deploy to the internet, and the rest of the team was able to utilize the new features associated with the API. Heroku has a free tier that the team was able to use, and in the future this free tier should support the development by the sponsor.

6.3.7 Comparing Web Frameworks

Once our MySQL database has been set up and the tables are all connected and filled as necessary, we must create a web server to allow our website and mobile application to “talk” to the database to get pertinent information. Traditionally, this was done with HTML and PHP, and more recently a GUI interface known as PhpMyAdmin. Php is also the language of choice by the project sponsor. However, in the spirit of Senior Design, our team would like to bring this project to life using more modern and sleek frameworks which are being quickly adopted by developers. For the website, research is being done into frameworks such as Angular.js and React.js. Commonly used alongside these frameworks are the backend frameworks Node.js and Express, which are based in Javascript. For the purpose of the web server research, we will be comparing a second option for back-end code using Flask, which is Python based, to the Node.js and Express pair. These two options are fairly similar, and for the most part, our project can be achieved by using either framework. None of the features of the

application require any specific feature of a framework, however there are differences in the development process involved with the two which can mean a great deal to the success of the team. [8]

Flask, being Python based, is fairly simple to learn and get a web server deployed with little experience required by the developer. As our team is inexperienced in regards to web development, Flask initially seems to be the better option. The Python language itself is potentially the closest to spoken english that programming will get in the next several years, which would mean explaining the back-end of CemetARy to anyone would be slightly easier. Flask is also a fast-to-deploy framework, as the Netguru blog mentions: “most experts agree that developing a Python application is 5 to 10 times faster than developing the same application in Java.” The issue with using Flask, however, is that it would require two different languages to be used in our web stack. [8]

The website framework of the project is planned to be in React.js, which is based on Javascript. Node.js and Express are based in Javascript, and experience with previous projects shows that React and Node are coded almost exactly the same, and the interconnectivity of the two is all but seamless. Another pro to using Node.js over Flask, according to Netguru, is the fact that Python is not completely supported through most browsers or mobile environments. Javascript, on the other hand, is the most widely used web development language by a large margin, and allows for easy interaction between the different browser environments. For example, Node.js has easy functions for adding, editing, and removing information directly to and from the browser’s session storage. In a previous project by Joshua, this was used as a work-around to tokens in a situation where the group was pressed for time. For the reasons above, the back-end framework of choice is Node.js and Express. This decision completes our web-stack as the MERN Stack:

M. MySQL

E: Express

R: React.js

N: Node.js [8]

6.3.8 Collecting navigation data

In order to collect the data points for each of the headstones, section perimeter points, road points, etc. We needed to have more accurate points than what a simple cell phone can provide. Dr. Giroux has provided us with a list of data points for the Greenwood cemetery; however, in the future we may need to collect more data points for stretch goals or testing. To accomplish this, Dr. Giroux has offered to lend us the use of her GPS Data collection device: The Garmin GPSMAP 64st. The GPSMAP is a handheld device containing a high sensitivity antenna, which allows for accurate GPS point collection regardless of the terrain or lack of signal in the user's phone. The user can simply click a button to collect the most relevant point near them, and then connect to a mobile device via Bluetooth to see or export the points. [9]

6.4 OCR Research

6.4.1 Meeting with a previous Senior Design team

During our fifteen minute presentation before spring break, Prof. Mark Heinrich asked a group in Senior Design 2 to meet with Jonathon. Meeting with Philip Rodriguez and Collin Speight was very helpful. They were able to give Jonathon a general idea of what the general process of OCR involves. They also provided a list of terms to research:

- Tess-Two

- Filter to binary image
- Open CV
- Character segmentation
- Tile image character
- Segmentation per word
- Transfer learning yolo
- Character detection
- Word detection
- Tiny yolo mobile net
- Ios metal
- Spellcheck results
- Cut text, binarize image, feed to tesseract
- Data augmentation
- Single shot detector
- Tensorflow light (object detection)
- Tesseract 4.0
- Find and magnify difference
- Niels Lobo
- Ulas Bagci
- Google's ML Kit

6.4.2 OpenCV Overview

OpenCV a.k.a. OpenComputerVision is, as the name implies, an Open-source Computer Vision library. OpenCV is widely used by seemingly every computer vision developer in the world. There are hundreds, possibly thousands, of articles and blogs dedicated to explaining how and why they used OpenCV. The focus of OpenCV is on real-time applications, perfect for a mobile OCR app. Read directly

from their website: “It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications.” [10]

OpenCV even has a built in OCR feature. However, OpenCV’s character recognition does not have the most stellar of reputations. That is why we decided to look into using a different library for the character recognition. The most widely used open source OCR libraries with the clearest and most plentiful documentation are currently Tess-Two and Tesseract 4.0. [10]

6.4.3 Tesseract Version Split

Tesseract has had a long development history. Originally, Tesseract was developed by Hewlett Packard as proprietary software. Hewlett Packard eventually released Tesseract as open source in 2005 and Google sponsored development in 2006. A few years later, in 2011, Robert Theis released a fork version called “tess-two”. Just in time too because just two years later in 2013 Tesseract development would slow until 2018 when Tesseract 4.0 would be developed. [11]

6.4.4 Tess-Two

Tess-Two is a fork of Tesseract’s Android toolkit. Tess-Two is a collection of Android APIs, build files and training data for the Tesseract OCR and the Leptonica image processing libraries. It contains a Java API for making use of natively-compiled Tesseract and Leptonica APIs. [12]

Tess-Two looked promising initially. It even has its own broad selection of training data to make use of. However, Tess-Two is exclusive to Android

development which is a rather large problem considering that one of our sponsor's requirements is that our app work on iPhone 7s. [12]

6.4.5 Tesseract 4.0

The previous versions of Tesseract 3.x.x were primarily focused on character patterns. Tesseract 4.0 is different because 4.0 focuses on line recognition using their new neural network LSTM. There is still a Legacy OCR Engine mode (--oem 0). Tesseract 4.0, like Tess-Two, also has a repository of training data for various needs. [13]

6.4.6 Image Processing

Similar to how OpenCV has a built-in OCR function, Tesseract 4.0 does have a image preprocessor. Also much like OpenCV's OCR, the image preprocessor does not have the greatest of reputations for anything that lies outside of the realm of scanned documents. As you can see by Figure 6.7, this means that we will need to do a lot of work preprocessing to maximize Tesseract's character recognition. [14]

Fortunately there are quite a few things we can do to help Tesseract work. Number one, resize images. All images that are passed from OpenCV to Tesseract will need to be at least 300 DPI. [14]

Secondly, binarization. All images that are passed from OpenCV to Tesseract will need to be black and white. Specifically, the text should be white and the background black. Tesseract does have its own implementation of Otsu's algorithm (more on binarization later), but Otsu's algorithm is limited to images of bimodal lighting. There are a few alternatives to Otsu's algorithm, namely:

ImageJ Auto Threshold, OpenCV's image thresholder, and scikit-image thresholding. [14]

Thirdly, noise reduction. Noise is defined as an image's random variation of brightness or color. Noise is not a challenge for the human eye. However, a computer algorithm can find noise to be a limiting factor of their accuracy. Tesseract, again, has a built-in process for noise removal, but again its process is limited. [14]

Fourth, image rotation. Unlike with properly scanned documents, when one of our users are taking a picture of a headstone there is no guarantee that they will hold their phone straight up. Tesseract 4.0's ability to use its line segmentation neural network is significantly impaired if the text is presented at an angle. We will need to address this by rotating the characters. Rotating the image is more common, however the headstones that we are dealing with often have text arching across the headstone. To accommodate this, we will need to detect the characters individually and cut and rotate them before passing them into Tesseract. [14]

Figure 6.7 Even binarized, there is still so much excess data that needs to be removed.

Fifth, cropping junk data. Anything in the background of an image can be falsely detected as extra characters. This problem is amplified when shadows are involved. Problematically, Tesseract also has difficulty with text that has no border at all. To account for this we will need to crop the characters, rotate them and then add a white border around them. [14]

Sixthly, single threading. Tesseract by default runs on four threads. This should not cause any issues for smartphones… just in case though. The Tesseract faq page declares that:

> "Using a single thread eliminates the computation overhead of multithreading and is also the best solution for processing lots of images by running one Tesseract process per CPU core.
>
> Set the maximum number of threads using the environment variable OMP_THREAD_LIMIT.
>
> To disable multithreading, use OMP_THREAD_LIMIT=1." [15]

We do not necessarily expect to need to know this, but if it turns out that this shaves off time from the execution we intend to test the difference between multithreading and single threading.

Seventh, page separators. Another potential issue are page separators. By default Tesseract terminates each page of text by the FF character. If page_separator is set to LF Tesseract would add an empty line the the end of its pages of text. We intend to feed in character by character so we will most likely need to set the page_separator to an empty string. [15]

Eighth, page segmentation method. As previously stated, Tesseract uses a line segmentation neural network to parse images. Since we will be passing a single character to Tesseract we will need to use a different segmentation mode. The segmentation mode can be set via the --psm argument. The following is a list of the different page segmentation methods for our future reference. We will most likely be using mode 10. [14]

0 Orientation and script detection (OSD) only.

1 Automatic page segmentation with OSD.

2 Automatic page segmentation, but no OSD, or OCR.

3 Fully automatic page segmentation, but no OSD. (Default)

4 Assume a single column of text of variable sizes.

5 Assume a single uniform block of vertically aligned text.

6 Assume a single uniform block of text.

7 Treat the image as a single text line.

8 Treat the image as a single word.

9 Treat the image as a single word in a circle.

10 Treat the image as a single character.

11 Sparse text. Find as much text as possible in no particular order.

12 Sparse text with OSD.

13 Raw line. Treat the image as a single text line, bypassing hacks that are Tesseract-specific. [14]

Nine: dictionaries, spell check, and number versus character. Since we plan to be using Tesseract to identify single characters, disabling the dictionaries Tesseract uses should reduce computational time. The dictionaries can be disabled by setting the configuration variables load_system_dawg and load_freq_dawg to false. [14]

Instead of dictionaries, we will be flagging the input as a character or a number. Tesseract can be set by using the tessedit_char_whitelist configuration variable. [14]

Once the characters or numbers have been identified we will need to arrange them for use in the database. This involves spell checking the names and adding ** to any dates that use the ‘XX year format.

6.4.7 OpenCV Binarization

As mentioned previously, all images passes into Tesseract must be binarized. OpenCV has three methods of binarization: Simple Thresholding, Adaptive Thresholding, and Otsu's Binarization. [16]

6.4.8 Simple Thresholding

Simple thresholding is simple: choose a threshold value and set each pixel to black or white depending on the pixel value relative to the threshold value. The OpenCV function is simply called cv2.threshold. Pass in a grayscale image, the threshold value, maxVal, and the threshold style. The maxVal decides whether above the threshold should make a pixel black or white. OpenCV has five different methods for handling the threshold. Mainly binary truncated and to_zero. Simple thresholding outputs a retval and the processed image. Retval is used in Otsu's binarization. [16]

6.4.9 Adaptive Thresholding

Simple thresholding uses a single threshold value. That can work fine for simple images. The headstones that we need to work on however are not so well lit that simple thresholding would work. The next best alternative is adaptive thresholding. Adaptive thresholding separates the image into several smaller sections and calculates a probable threshold value for each section. This is a large improvement for real world lighting that changes over different parts of the headstones. [16]

Adaptive Thresholding has three input parameters and a single output image. The first parameter is the adaptive method. The adaptive method determines which algorithm is used to calculate the threshold value for each region. There are two algorithms to choose from, the mean and gaussian methods. The

gaussian method uses a weighted sum. The sum is weighted by gaussian windows. The gaussian window is a math concept that we are unfamiliar with. It seems to take in a bell curve of values and output a threshold. [16]

The second parameter determines the size of the quadrants of the image. The smaller the sections are the better it is, to a certain extent of course, the sections can only be so small before they become impractical. The third parameter is a simple constant that is subtracted from the result of the mean and gaussian method. [16]

6.4.10 Otsu's Binarization

Tesseract's built-in binarization method uses Otsu's binarization. OpenCV has its own implementation as well. There is a lot math behind the scenes, but put simply Otsu's method takes an image and analyses its histogram and chooses a threshold value between two peaks of the histogram. This means that a noisy image that does not have a distinct bimodal histogram will not work out great. Thankfully there are ways to reduce the noise in an image with a gaussian kernel. [16]

OpenCV uses the same function as simple thresholding for Otsu's binarization. The only change is that the threshold value starts at zero and you need to pass the cv2.THRESH_OTSU flag. [16]

These are code bits for future reference. A five by five gaussian filter and Otsu's binarization in OpenCV. [16]

```
blur = cv2.GaussianBlur(img,(5,5),0)
ret3,th3 =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

6.4.11 OpenCV Character Detection

When Jonathon was meeting with Philip Rodriguez and Collin Speight they emphasized that for our intended design, we will need to be able to locate where the characters are. We cannot simply throw the entire image at Tesseract and use whatever it outputs. For our design we intend to have the user classify the input as a name or date. This should greatly increase the speed and accuracy of the OCR and minimize user frustration. However, in order to do this we need to be able to isolate the characters in whatever area of the text that the user highlights. [17]

To this end, we found an informative blog post from Dan Vanderkam's project for the New York Public Library. He was looking to crop down and isolate the small amount of text in a large image. What follows are the highlights of the relevant parts of his findings as they apply to our project. [17]

The first thing Dan Vanderkam does, and the first thing we will do as well, is to use the canny edge detector on the input image. As the before (Figure 6.8) and after (Figure 6.9)images below demonstrate, the canny edge detector places white pixels wherever the algorithm detects an edge. [17]

Figure 6.8 Input image before the canny edge detector is applied

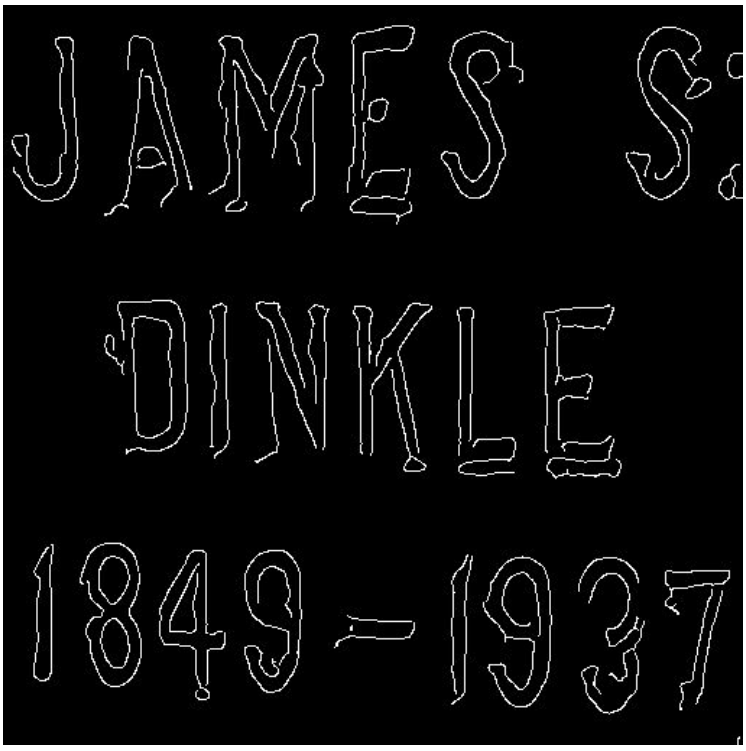

Figure 6.9 Input image after the canny edge detector is applied

As you can see, the algorithm does a pretty good job of cleaning most of the excess data from the image. However, the font used on the tombstone is super bold. This produces, in equal parts, extra lines and extra problems.

In order to have clean data of the database to search we will need to clear out the extra bits on the edges. Dan Vanderkam's input image did not have quite so much noise, but we will discuss solutions for this in a separate part. [17]

At this point Dan Vanderkam had some borders and smudges to deal with. His process for clearing out the borders was clever but, unfortunately, not directly applicable to our project. [17]

After removing the border Dan Vanderkam explains his next step as a crop of four points that has as much of the white edges as possible. However, the crop also needs to be small enough that and extra data is left out. [17]

Maximizing the white pixel edges would mean taking almost all, if not the entire image. This is clearly unhelpful. Thankfully, Dan Vanderkam is more experienced than us and identifies this dilemma as a classic precision/recall tradeoff. In this case the recall is the fraction of white edge pixels that we keep and the precision is the fraction of the white edge pixels that we discard. [17]

Dan Vanderkam solves this precision/recall problems by optimizing for the F1 score. The F1 score represents the harmonic mean of precision and recall. The harmonic mean is defined as: (2* precision * recall)/(precision + recall). [17]

As you can see from the above image (Figure 6.9), the edges are not a contiguous character. Before optimizing for the harmonic mean, we need to

ensure that the characters we optimize for are not orphaned. Dan Vanderkam uses binary dilation on the edge image. Binary dilation bleeds the white edge pixels into each other. As shown in Figure 6.10, if this process is repeated enough times and the characters can become quite distinct.

Figure 6.10 Input image after canny edge detection and binary dilation are applied.

This image (Figure 6.10), is a rough proof of concept demonstrating how we could use binary dilation.

Dan Vanderkam uses binary dilation quite a few more times to make the entire block of text a single contiguous block. However, we are identifying characters instead of sections of text. So our use will be slightly different. [17]

The next step is identifying the crop. A.k.a. Maximizing the F1 score. Dan Vanderkam uses a greedy expanding crop on each block of white dilated edge pixels. Expand until the harmonic mean stops increasing. [17]

For our input however, the character are too close together to use a four point crop. We suspect that we will need to use a type of flood crop. More research needs to be done into the mechanics of this operation.

6.4.12 Initial Problems

As far as initial testing goes, we have tried many ways to remove noise from an image. Edge detection and dilation is the most promising, however some headstone are extremely noisy. For example, this headstone (Figure 6.11) looks innocent right? Wrong.

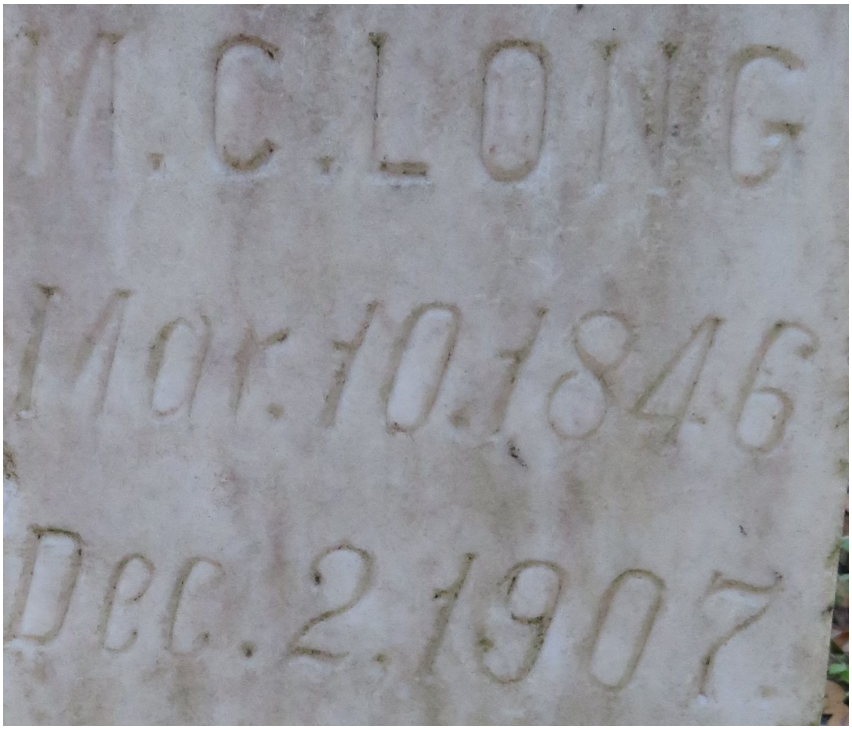

Figure 6.11 This headstone is secretly a problem child.

As you can see in Figure 6.12, so many characters are totally falling apart or split into pieces.

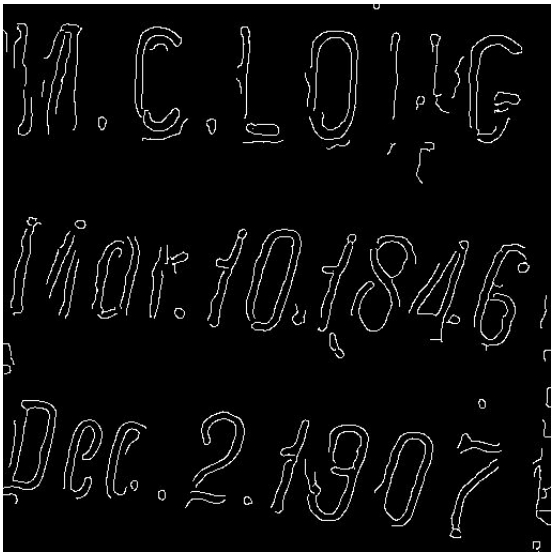

Figure 6.12 The secrets of the bad headstone revealed!

We also tried to use a local normalization method to reduce noise (Figure 6.13), but it did not help at all.

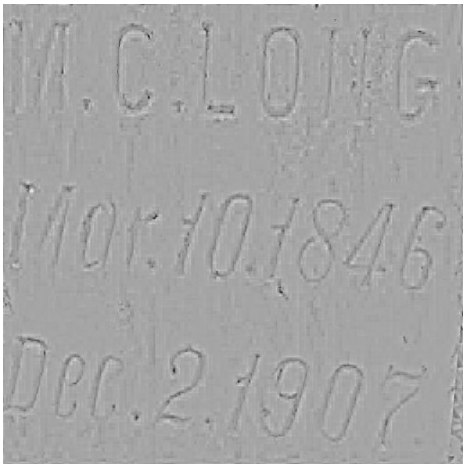

Figure 6.13 We tried so many methods to clean the noise, but in the end it did not even matter.

This issue may be unresolvable without a better source image. The headstone used in the “OpenCV Character Detection” section worked out much better. Initial testing showed promise with Tesseract being able to read the name and date mostly right even without any custom settings. This kind of inconsistency could

be a big problem for the actual user experience however. See the Preliminary Design for OCR Process for our proposed solution to this issue.

6.5 Front End Framework Research

During the process of researching which framework to use to develop the front end of the website, we have found that React is one of the most popular and is used by several different reputable companies. Our web developer has had a good amount of experience with creating website user interfaces using just the standard HTML and CSS, but we feel that utilizing a framework such as React is much more appropriate for this project. They do have some experience working with JavaScript so with the fact that React is an open-source JavaScript library, we were be able to learn the syntax relatively quickly and implement it on the initial website that has been created.

6.5.1 React vs. Angular

We have also taken a look at the Angular framework because from what we have found Angular is one of Reacts biggest competitors when it comes to the most searched and popular front-end frameworks that are used as of this year. The issue that we might have had with choosing angular is that it involves using a scripting language known as TypeScript. From what we have researched, TypeScript is meant to be a superset of the language JavaScript. Therefore, although our web developer has some knowledge of JavaScript, the learning curve for learning Angular would have probably taken longer for them to familiarize themselves with than leaning React which requires the use of only the JavaScript language. [18]

The main reason for using a framework like React for the user interface is that it allows the reuse of the UI components. React provides the ability to create

several function-like pieces of code known as “components”. Each component represents a different part of a user interface. The benefit that React provides is the ability to reuse these components throughout several webpages. This functionality is a great way to save time during the development process. It keeps all of the components separated from each other so that changes made in one component does not affect the others. [19]

The problem when only using HTML for a web page is that every time that data is changed on the page, the webpage constantly needs to be reloaded on the browser. This makes things run much slower because the browser has to continue doing more work before the next change can be made. React provides the ability for data to be changed on webpages and the browser does not need to reload the pages after every change. This not only helps on the browser side, but also provides the user with a more pleasing and faster user experience.

Another reason that we were a bit more interested in using React rather than another framework is because it uses the Virtual DOM which is an abstract form of the Real DOM. Using JavaScript to render the HTML makes it easy for React to keep a virtual representation of the HTML in memory. Other frameworks rely on enhancing HTML to make it work with data. This makes it easier to update changes that the user has made without it having an effect on the other parts of the user interface. All of these reasons make developing a React application efficient and saves lots of time during the development process. [19]

React uses JSX which is simple JavaScript that allows for the use of HTML within it which is processed within the JavaScript. This is useful because we were able to use the HTML that was initially created for the home page and sign up forms for the website. Also, we do have some prior experience using JavaScript to create web applications and from what we have researched a developer with

knowledge of JavaScript could become a productive React developer in a matter of hours. Learning and utilizing React would not only be easier because we have prior knowledge of how to use JavaScript, but also because from the research that we have gathered comparing it to other frameworks, React seems to have the shorter learning curve overall. [19]

Another benefit is that React is open source, which provides free access from several other applications and additional tools to utilize that have been created by other developers. For example, React is number 5 in trending on GitHub and it currently has over 1200 open source contributors working on the library. New tools and features are constantly being developed by developers that are not a part of the company. Some of Reacts main features were created by developers that saw the open source project on github. [19]

React also has great tools that we can use to debug the system during the development process. It has a dedicated debugging system that can isolate certain user interface errors and bugs that will help find the exact component that might be causing the errors. Aside from that, the browser also will provide useful information about the specific lines of code that are causing the website to have errors. Google Chrome's developer tools help lead the developer to find the exact section that is causing the issue so that they can make the necessary adjustments to correct the problem. This is most definitely useful and made the development process of creating the website much easier. [19]

6.5.2 Potential Disadvantages

Equally important when researching a framework to utilize when developing a website, is not only to research the benefits that it provides, but also to find out

some of the downsides to the framework. There is a reason that there are several frameworks out there and certain companies choose to use one over the others. Each framework has its advantages and its disadvantages when compared to its competitors. If there was a framework that exists that has no downsides to using, then there would be no reason to have several frameworks and everyone would just use the same one.

One of the disadvantages that has been found about React is that it's lacking when it comes to useful documentation. React is constantly releasing new tools at a fast pace. We have read about developers hitting a roadblock during the development process while using React. For example, when trying to combine React with other useful libraries that the developer would like to use, there is little to no information on how to do this. If they are lucky and they do happen to find some information on the topic, often times the information is not useful.

When compared to Angular, which has been around for a longer period of time, there are now several resources and tutorials at the developer's disposal to help them learn how to use all of its features. It also seems to be continually changing at a rapid pace. This could be a good and a bad thing. The obvious upside to this is that its continuing to grow and improve quickly. [20]

Another downside might be that React seems to be dependent on other external libraries when it comes to adding more functionality to the website. Apparently, there is only so much that its core functionalities can accomplish to enhance a web application. Having to continually add and learn new libraries could definitely become a hassle over time. Also, maintaining several libraries all at the same time could be more complicated than it needs to be.

6.5.3 Final Decision

Overall, it seems that React certainly has several benefits. From all of the research that has been conducted so far it seems that we are finding more resources that state its benefits and relative to those benefits. We are finding few downsides. There must be a reason that React has continually been able to remain relevant on the market for several years. Companies are still continuing to invest into this technology which gives us more confidence that it will be useful and continue to grow for years to come. As all other frameworks, it has its downsides and its benefits. The overall consensus seems to be that it is a useful library for creating modern web applications.

The research into React continued as we implemented it into the project more and more. We began by adding some of its functionalities to the initial website that has already been created using technologies including HTML, CSS, and the Bootstrap framework. As we went through the development process, we continued to learn more and more about this framework which we found to be exciting and beneficial for our skills as a developer. From the research that we had done at the start of this project It seemed that React would be a great option to use for what is required of this project.

It really was a hard decision when it comes to deciding between the current top two frameworks Angular and React. Although, we ended up using React because of all of the reasons we have previously stated above. If we did end up having a need for some of Angular's functionalities, it has been found that it is possible to integrate some of Angular's libraries within our React project.

It was also possible as we continued to develop this website with React that it might not have been the correct fit for this project and that Angular might have

been the right way to go after all. This is fine because then at least we would gain some experience with both technologies and we will have a better idea of when and why to choose one framework over the other. This would only have been a benefit to our overall skills as a developer. This entire project was meant to be a beneficial learning experience so even in the case that we did need to change frameworks it wouldn't have been taken as a bad thing whatsoever.

7. Project Design

7.1 Website Design

7.1.1 Website front-end Design

This section is intended to provide an overview of the software design of the front-end website of the system including the technologies used as well as the languages and different software implemented to create the user interface of the website. The website portion of the project is intended to provide functionality for the administrative users to update and maintain the information that is stored in the back-end database of the overall system.

This includes web pages with forms provided for the user to enter the relevant information on the veterans, some of which are the veterans' names, their rank, awards that they may have received, serial number, date of birth, date of death, etc. There will also be forms provided for the administrative users to enter the GPS coordinates of their cemeteries including the individual sections as well as the coordinates of each of the veterans headstones that will be used for navigation.

The initial goal for the design of the website front-end was to create a clean and user-friendly dashboard (Figure 7.1) that provides navigation to the various pages of the website. It includes a picture of the veteran cemetery and a navigation bar at the top that provides the user easy navigation to the sign in page (Figure 7.2) where the administrator will need an account to login to the website. It is a vetted login page that will ensure that only authorized users will be able to successfully log in.

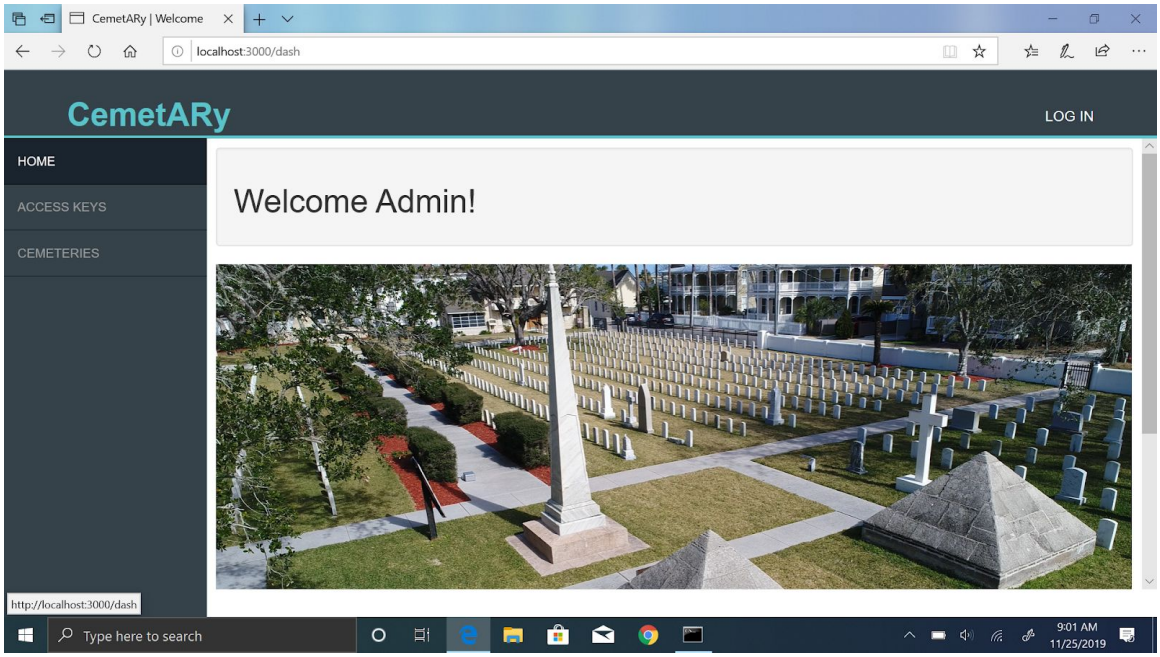

Figure 7.1 Home page. This is a sample of the home page that users will see when log into the website.

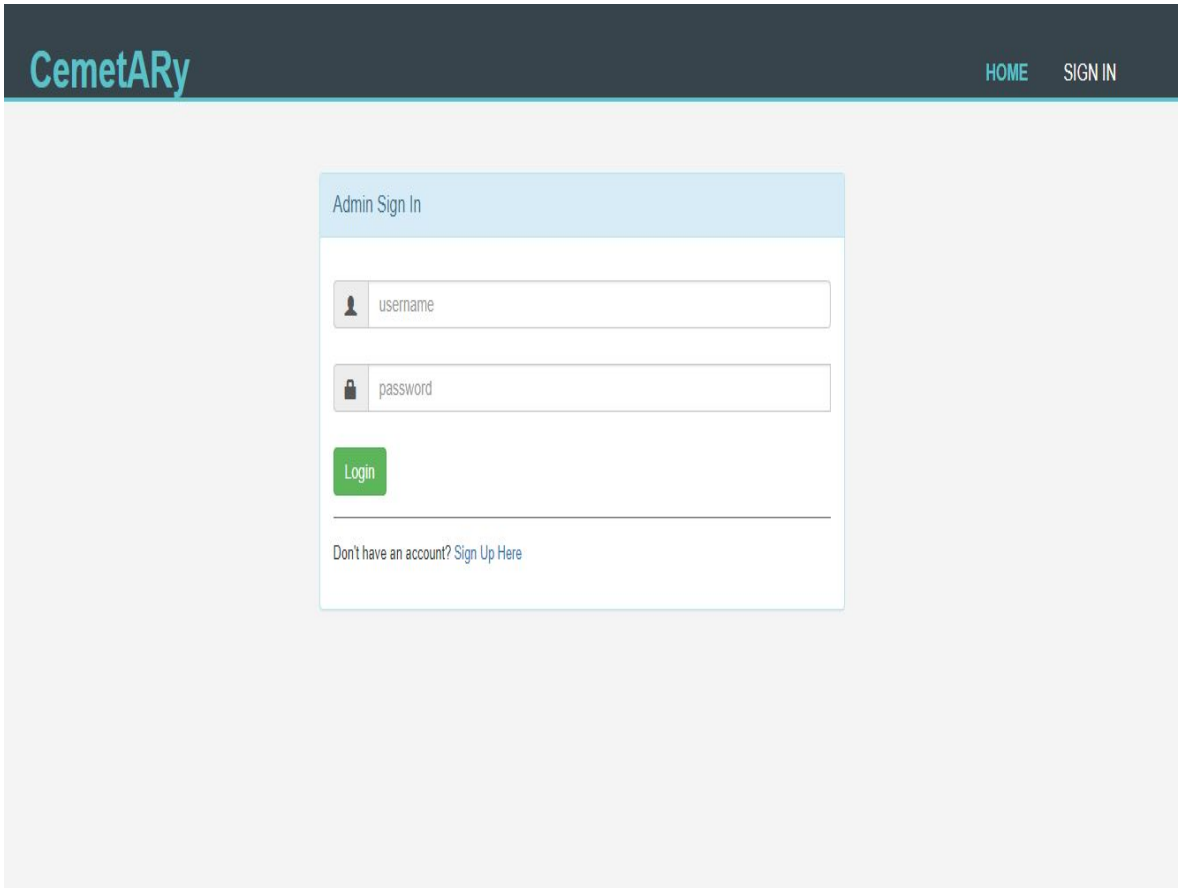

Figure 7.2 Sign in Form. This is the sign in form for admin users to sign into their account.

The sign in page provides functionality for the user to login to an existing account that has been stored in the database and will be checked before providing the user access to the database update and maintenance web page of the site. If the user does not already have an account, there is an option at the bottom of the sign in form provided for the user to sign up for a new account which will take the user to the sign-up form (Figure 7.3). This form will require the user to provide their personal information that will be stored in the database.

This information includes the user's first and last name, personal email address, and desired password. They will also need to enter an access code that has been provided by an administrative user that gives them permission to create an

account. The access code is the solution our team and our sponsor have agreed on regarding the security of a new user creating an account on the system. Only approved users that have been provided with an access code will have the ability to sign into the website and have access to the database information update and maintenance page of the website.

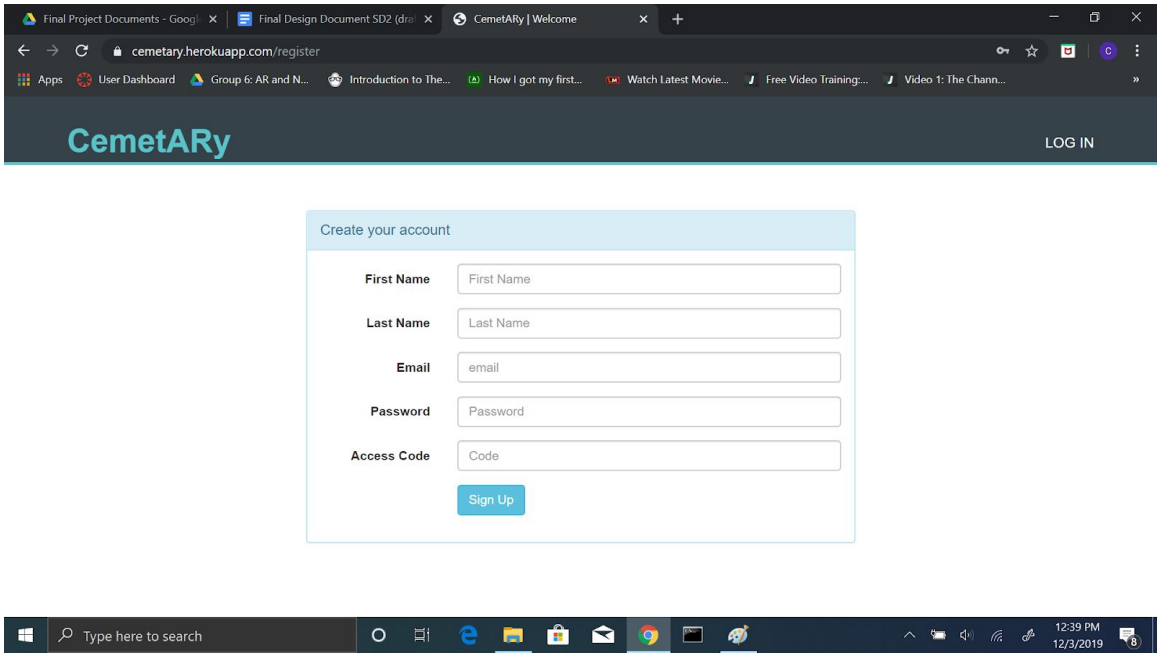

Figure 7.3. This is the form for a new user to create an account requiring an access code.

7.1.2 Form Design

The design of the forms on the website were created by utilizing the web development framework React as well as the programming languages HTML, JSX, JavaScript, and CSS. React was used to create the overall content and frame of the forms. It was used to set the margins of the forms as well as provides the text and input boxes where the user will type in the necessary information. React was also used to provide the functionality for the buttons on the form. For example, in the event of a mouse click on the login button, the code within the JSX will call a function that we have created what is called

“submitHandler”, this function works as follows. It will retrieve the information that the user has entered and then it will check if that username and password combination is currently stored in the database.

If the algorithm is successful in finding a match to the users input then they will be granted access and be logged in and the user will then have the ability to use the rest of the intended functionality of the website including entering or updating information on the veterans into the database. If the username and password combination data is not found in the users table of the database, then the user will not be able to log in and they will remain at the login screen where they will be able to attempt to login again just in case the user accidentally made a mistake entering their information such as a misspelling their username and/or password.

The sign-up button located at the bottom of the sign-up form has similar functionality to the login button where it will also call a function on a mouse click event which will then gather the information that is currently in the input fields. It will first check the access code field that the user has entered with the access code that is stored in the database. If the access code is a match, then the other information on the form that the user has entered will be sent and entered into the administrators table of the database. The user will then have successfully created an account on the website. After an account has been made, the user will then be sent back to the sign in form where the user will have to go through the sign in process on the sign in form that has already been described above.

7.1.3 Access Code

Regarding how the access codes will be generated, our sponsor will have the functionality to generate them directly from the website. There is a button

provided that will generate new codes when it is clicked on. The codes will be stored into the database so that they can be checked when a new user inputs a code when attempting to create an account. The codes will initially be stored in the database with an attribute that they are currently inactive. Once a new user creates an account using an access code, the attribute for that access code will be changed to currently active. Once the codes have been generated they will then be viewable in tables provided so that they can be distributed to the new users that have been approved access to create accounts on the website. There is also the option to view the active keys along with the administrators information who used those keys so that users know who owns what key and can contact them if needed.

With the initial design completed for the functionality of allowing users to create new accounts as well as signing them into their accounts, we next needed to approach the functionality provided to the administrative users that successfully log into their accounts. It is important to note that this project will be applied not only to the Greenwood Cemetery, but also to national cemeteries all over the United States. So, the idea is to allow users to create an entire new cemetery object that can store all of the relevant information on that specific cemetery. The first user interface that the user will see after logging in to the website will show the dashboard of the website, welcoming the administrator with a nice picture of the Greenwood Cemetery. There is a navigation bar on the left side of each page to provide easy access to navigate to all of the pages within the website. The third navigation tab will navigate the user to the cemeteries page where all cemetery information can be edited. There are tables provided to view all of the cemeteries within the database as well as their sections and all of the veterans within the database. This will allow the user to easily look through all of the existing cemeteries and help them immediately navigate to section containing their cemetery that they would like to access and update information on.

There is also the option to create a new cemetery within the system. The user interface for this will be a table of the cemeteries that have already been added and at the top of the table there is a button provided that will allow the user to add a new cemetery. This will open a form for the user to enter the information of the cemetery. The user will need to input the name of the cemetery and the city that it is located in and when submitted it will then be added to the list of cemeteries. The cemetery will also be added into the database so that the user can immediately input the other information that they would like to enter into the database. Once a cemetery has been added to the database, it will be given a unique ID. This ID will be used when adding information into the database for a specific cemetery. For example, when a user would like to input a section within the cemetery they will need to provide all of the information on the section as well as the ID of the cemetery so that the database knows which cemetery to store the section under. There is similar functionality for adding GPS points within sections where the user will need to provide the ID of the section that the point is located.

Apart from the content of the webpages, the website would look plain and unappealing without the utilization of cascading style sheets or CSS. CSS was used to add some color and style to the webpages and helps align the content on the page in a structured way. It is convenient because the same style sheet can apply to multiple React components that we created which keeps the theme of each of the webpages consistent and helps the entire website flow smoothly. The design also includes the use of the Bootstrap framework to create the responsive and visually pleasing sign in and sign up forms. We have found that the regular forms that you can make using just JSX and CSS can be plain and not visually pleasing. By utilizing Bootstrap in the JSX source code, all of the input forms on

the website now have a more professional and cleaner look that is more on par with today's standards of web development.

7.2 Database Design

7.2.1 Database Structure

Once the database had been set up on Heroku, we created all of the tables necessary to keep track of all of the information. There are several tables that must be included that associate with the veterans, but we must also create a table for the administrator users detailed in section 5.3. Dr. Giroux provided the structure that the Veterans Legacy Project has in place, and our application will follow that structure closely, if not exactly. The following Entity Relationship Diagram in Figure 7.4 gives a visual description of the tables provided and their associated auxiliary tables.

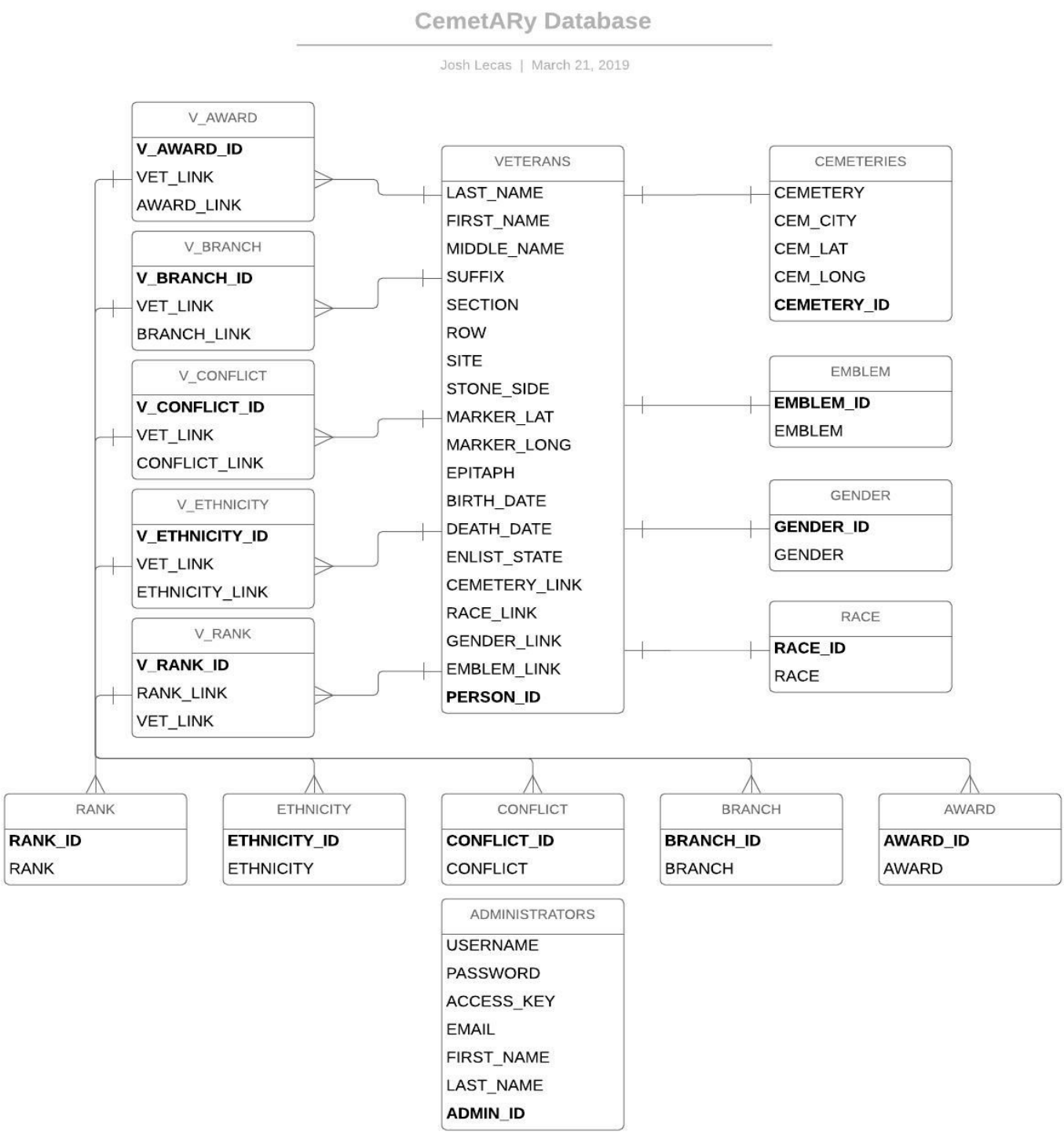

Figure 7.4 Initial Database Schema

The Entity Relationship Diagram above requires an explanation, as it may not be fairly clear upon first inspection. This particular diagram follows the schema as provided by the sponsor, with the addition of an administrators table to contain the information pertinent to the management of the website. Another feature that the current diagram is lacking is a sections table and a roads table. These tables will be used to store the GPS data points to be used with the navigation portion of the application. The roads table however, is part of a stretch goal of navigation and was prioritized lower than getting a sections table included.

There are several tables included with the diagram that are not the administrator or veterans tables. Not all of the information that we need to store about the veteran can be stored in a single table. We have also included four tables directly related to the veterans table. These tables are the cemetery, race, emblem (religion), and gender tables, which are all direct children of the veterans table. Any direct children of the main table are associated by a one to one relationship line. The tables on the left hand side of the veterans table are auxiliary tables to allow for many to many relationships.

In order to allow for many to many relationships in a relational database, there must be auxiliary tables which connect one table to another by retaining the primary key pair for each table. For example, one of the tables in our database contains all of the different conflicts which can be related to the veterans buried in the numerous cemeteries. There is a high chance that a specific veteran has served in more than one of these conflicts. In fact, in our Greenwood cemetery testing bed there are several veterans who have been involved in a number of conflicts. For each conflict that a veteran was a part of, there is a primary key pair included in the auxiliary table V_CONFLICT. The database, and eventually the API will be able to connect these tables together by using these pairs of primary keys.

For the two tables ethnicity and rank, these tables initially appear to be implementable as a single column in the veteran table. However, according to the Dr. Giroux, history students involved with the Veterans Legacy Project have discovered that many of the veterans buried are from different countries, and are of different ethnic backgrounds. The ethnicity table allows us to add new ethnicities as the veterans are studied, as well as account for veterans who may be of multiple ethnic backgrounds. The rank table is necessary as the veterans who served in multiple branches usually held more than one rank. A many to many relationships accounts for thisl.

Conversely, the administrator table does not have any relation to the other tables in the diagram. The table is included in the diagram to show that it is in fact included inside our database but is separate from the portion of the database that will be accessed by the mobile application.

7.2.2 Additional Required Tables

Aside from the initial schema that the sponsor provided, modifications are necessary in order to complete our goals. In designing the database, we realized that it is highly necessary to add more tables and relationships. The first example of this is the requirement for an access key to be used when registering for the website. A feature of the website is the ability to view all of the active access codes and their statuses and generate new codes from directly within the dashboard. In order to do this we must store the codes in their own table, complete with a primary key. A visual description of this table is in Figure 7.5

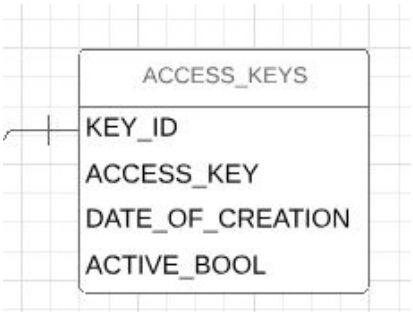

Figure 7.5 Access Key Table

This table has a one to one relationship with the administrators table. Also included with the access key is the date that the key was generated. This will help the administrator keep better track of the access keys. Also included in this table is a boolean indicating the active status of the code: whether or not they have been used already to create an administrator account. The intent behind this is to use the amount of active keys as a limit. For example only allowing a small number of keys to be active at one time, to increase the security of the website. The “active” status will change when a code is used, shifting from false to true. The active status is permanent.

A second table pair which is needed for our project is a Sections table and a perimeter table. Working with GPS coordinates in MySQL can be a tricky process, so these tables serve as a way to mitigate the pain involved. In the desired design, each cemetery will have a collection of sections, which contain each headstone. The navigation portion of the app will use these sections as an estimate of where a headstone is located to help users get navigated to a section, and then the app will refine the navigation down to the headstone itself. The actual structure of these tables is what makes working with sections difficult. Figure 7.6 is a visual of the structure of these tables.

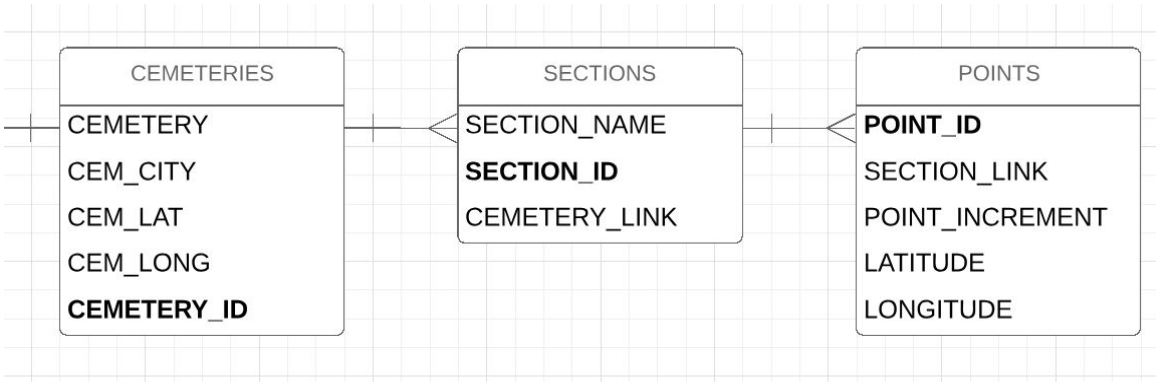

Figure 7.6 Section Table Structure

Due to the limitations of storing lists of objects inside tables with MySQL, the database must use a clever strategy to accommodate for the several GPS points which define a section. Each cemetery will have a one-to-many relationship with the new sections table, which will have a one-to-many link with a points table. This table will contain a long list of points which will be collected around important angles of the sections. As the points are added into the database, the cemetery will be selected, and the points will be incremented according to the time they were recorded. This way each section will have 3+ points which can be mathematically processed to create a 2d plane in which headstones are located. The purpose of this table is to allow a two-fold navigation. With a collection of sections, an API endpoint can give the user a point to navigate between sections, until refining the navigation down to a headstone. This allows for much easier navigation.

The points table ended up being a stretch goal for the project, as the sponsor was pleased with the way the application conducted navigation, yet has been included in this documentation and code to provide future developers with the ability to use these if necessary for the future.

7.2.3 API Endpoints

Below are the API endpoints used to communicate with the database. These endpoints may change in the future as features are developed after deployment, but these are the base requirements for an efficient application. Express offers middleware for routing which allows for a developer to easily separate API definitions into different folders and files. This results in the API development process taking a slightly longer time, but is a much more understandable organization. For the purposes of our application, the endpoint files will be split into three major sections: Administration, which handles all database admin functions such as access keys and authentication, data retrieval, which contains the endpoints for accessing information stored in the database, and data entry, which handles inputting and deleting information. As with all web-server development, certain API calls may need to be added, or are redundant and removed by the end of the project, luckily Express routing makes it quite simple to find and update these calls. Below is the list of files and explanations for each of their API Endpoints.

ADMINS.JS

The following endpoints require the use of JSON tokens, aside from the login endpoint, which will provide the user with the appropriate token.

Register - POST

The register endpoint makes use of the access key table, and will require the website to send registration data containing the fields detailed in the schema diagram. The endpoint will first check the access key against the table and check to see if it is currently active, and if not the password will be encrypted (Section 7.2.3) and the admin will be created. After the admin is created, another query is triggered which will update the consumed access key to “active”. If there is an

issue with the access key or admin creation, the endpoint will return an error code.

Login - POST

This endpoint will receive a username and password from the JSON payload, decrypt (Section 7.2.4) the password, and check if the user exists and the passwords match. If all the checks pass, the endpoint generates a JWT token for authentication, and returns it to the web page.

GenerateKey - POST

Simply put, this endpoint will generate a randomized key of a specified length, add it to the table of keys with an ‘inactive’ status.

List Keys - POST

To be used for maintenance, this will allow the website to display all of the information of the keys, for either distribution purposes or checking which keys are active. In production this section consists of two endpoints, one which returns active keys with the administrator information and one that returns inactive keys.

VETENTRY.JS

The endpoints associated with this file all require a token to be included along with the request, which is provided to the user through using the login endpoint specified above.

NewVet - POST

This endpoint allows for the dynamic creation of a veteran object and enters the veteran’s information into multiple tables at once, depending on the information provided in JSON. For the veterans table, the endpoint first builds several arrays of tags based on the information in JSON, if a certain field is not empty, its name

and information gets added to an array to be used in building the SQL query. Once all of the fields have been checked, the string is created and the query is sent out. After the initial query is sent out, several other fields are checked such as conflict, ethnicity, etc. These tables contain many to many relationships, and must be edited with seperate queries. These queries are only built and sent out if the original veteran insert was successful, to prevent errors.

NewCemetery - POST

AddCemetery will be seldom used after the application is built out, however for setting up the database it can make the process less painful. In the same vein as NewVeteran and NewSection, this endpoint receives data in JSON containing the required fields for the cemetery and adds it to the database.

NewSection - POST

NewSection works similarly to NewCemetery, but it also receives the id of a cemetery for which the section is associated, allowing the user to search for sections by cemetery.

NewPoint - POST

NewPoint works in the exact same way as newsection, however it receives the id of the section for which it is associated, as well as the point's latitude and longitude, and an increment variable, if necessary for development. For the final version of our project, the point endpoints were not used, but are included in case further feature expansion happens to subsequent versions of the project.

Deletion Endpoints - POST

Each of the above endpoints is paired with a deletion endpoint, which simply asks for the object's ID from the database, and then subsequently deletes the object.

VETINFO.JS

The endpoints contained in this file do not require a token to be passed along in JSON, as they are simply returning information and not directly interacting with the database.

SearchForVet - POST

SearchForVet is one of the more involved endpoints in the project. This endpoint dynamically builds a SQL select query based on filled out fields which are sent to the server in JSON. It utilizes the LIKE SQL keyword in order to allow the user to only scan partial sections of a name, which allows for a much more accurate search, and accounts for potential user error while scanning. Depending on the field included in the JSON request, the endpoint adds sections of SQL to a string to be executed. This handles tables like CONFLICT which have a relationship table in between it and the veterans table. In the future, more fields can be easily added by simply copy pasting if statements in code that handle the same relationship structure as the table desired to be added. Once the query string is built, it is executed and the list of veterans is returned in order of last name. The endpoint returns all of the information that a veteran has, and therefore a second endpoint for gathering location data about a veteran is not necessary as was initially planned, as the application can extract that information using this endpoint.

ViewAllVets - GET

This endpoint simply returns a list of all veterans, and is used by the website in order to display all of the veterans within the database. NOTE: This endpoint requires that the veteran has a cemetery and section associated with them, otherwise they will not be included in the results. This is the result of a feature included with the endpoint which allows it to return the cemetery and section that the veteran is inside, allowing for ease of use with the website.

Cemetery Information Endpoints

There are three endpoints associated with retrieving information about a cemetery. ViewAllCemeteries, ViewCemeterySections, and GetSectionPoints. The first endpoint returns a list of all of the cemeteries, as well as the information about each in the table. The section endpoint is a POST, and requires the ID of the cemetery for which the user would like to get the sections of. The points endpoint operates in the same way, but requires a section ID of the section associated with the points desired.

General Information Endpoints

There are nearly a dozen endpoints which are GET requests which return a list of the data in the associated tables. These endpoints are incredibly helpful to the website and the app as they allow the website to create dropdown boxes to "stupid-proof" entry into the database. They gather data associated with the additional tables to prevent the user from misspelling the names of certain data and causing issues in the database. For example, instead of typing in the conflict a veteran was involved in, a user can simply select a conflict which is already created in the database.

Future Implementation

A feature that was not able to be implemented was the ability to upload an excel spreadsheet of a certain format, and then parse through that data to easily upload a selection of veterans to the database. While the technology is out there, the team was not able to get this feature implemented in a short enough time, and was lowered in priority for the remainder of the project.

A second feature which is planned to be implemented is the ability to add information to the secondary tables such as Race, Conflict, etc.

7.2.4 Password Encryption

Passwords should never be saved into a database in plaintext. To avoid this, administration endpoints for logging in and registering will employ the use of Node.js libraries for encryption to create a hashed password, a “hashword” if you will, and save the encrypted string into the database. For logging in, the endpoint will take the plaintext password from the request, create the hash again, and compare it to the one stored in the database. There are two big libraries for encryption, BCrypt and Crypto. There is a problem with using Crypto however. Crypto relies on SHA256 encryption, which is good for processing large amounts of data quickly. The issue with quick processing is that modern servers can crack these passwords relatively quickly with enough hardware powering it.

BCrypt solves this issue by being incredibly slow to process. BCrypt introduces a work factor into the processing time, to prevent potential intruders from brute forcing the password. [21] The implementation of BCrypt can be tricky, as the speed of encryption can cause an issue with the API, as it is an asynchronous response, the password may not be encrypted before the request is sent to the database, causing an error. In pseudocode, the process for encryption is as follows:

```
bcrypt.hash( password ) {
        // Add user to database
}
// To login
bcrypt.compare(password,hashword){
if(match){// Login}
}
```

This ensures that the encryption will occur before the database is queried, assuring error free authentication.

7.2.5 Web Tokens

In order to ensure that the endpoints communicating with the database are secure and are only being used by a registered administrator, our back-end server will make use of authenticating web service tokens. One of the most popular implementations of this is JWT, or JSON web tokens. With JWT, a token will be generated upon logging into the administrator portal, which will then be authenticated before any of the endpoints can perform their queries. If the token is invalid, the intruder cannot access the database.

7.2.6 Environment Variables

In order to maintain security of the database, the API utilizes Environment Variables located on the team's personal computers, as well as the Environment Variables located on Heroku. These variables include the password for the database, as well as the secret key needed for JWT token generation. In the code, the API will first check to see if the Environment Variables from Heroku are able to be accessed, if not, it will attempt to load the .env file located in the project file structure on the local machine. This file is not included in the github repository, therefore preventing malicious attacks from happening to the database or project.

7.3 Application Design

7.3.1 Application Front-End Design

This section will describe the design of the front-end for the application. The user interface of the front-end was designed in Unity. When the users open the application they will see the Home Page, which will show the Main Menu options, as shown below in Figure 7.7.

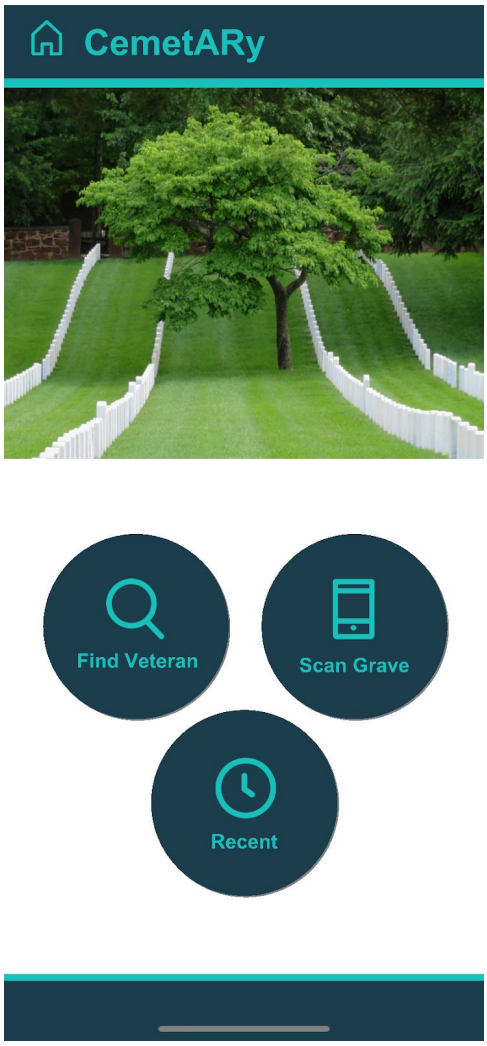

Figure 7.7 Homepage in Application

The first button on the screen, Find Veteran, allows users to search the database for a veteran. The second button, Scan Grave, allows the user to scan a headstone and find veteran information from that scan. The third button, Recent, allows the user to view recently scanned or searched veterans so that they can easily find them again. The picture in the background is a rotating set of pictures of cemeteries the sponsor provided to us.

The Recent button leads the user to a screen showing a list of veterans that they recently viewed the biographies of. The user can click a veteran and a biography of them will appear, as shown under the Find Veteran information below. We implemented this so that the user can easily find previous veterans they searched if they need to reference them again in their research. This list is persistent between loads of the application, saving the last twenty veterans viewed.

Figure 7.8 Recent Screen View in Application

If the user clicks Find Veteran it will lead them to a new screen which will allow them to search the veteran by fields that are stored in our database, including

name, location, states served, and conflict. The user can search by some or all of these fields, and can even input a partial name into the text boxes. It is then populated with a list of veterans that match that description below the search bar, as shown in Figure 7.9.

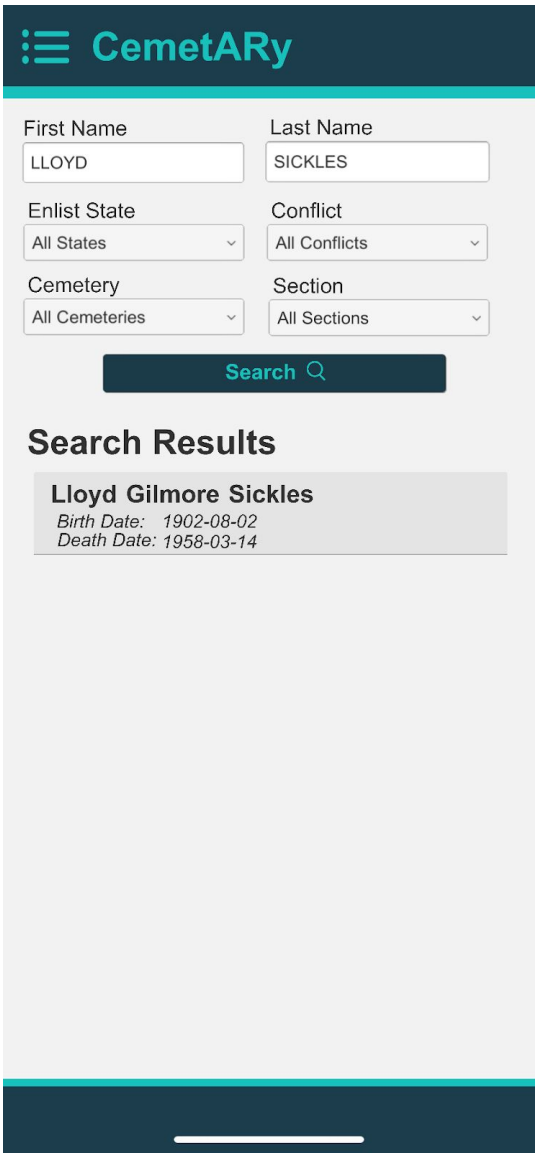

Figure 7.9 Search Screen in Application

If the user clicks a veteran, it will lead them to a new screen on the biography of that veteran, as shown below in Figure 7.10.

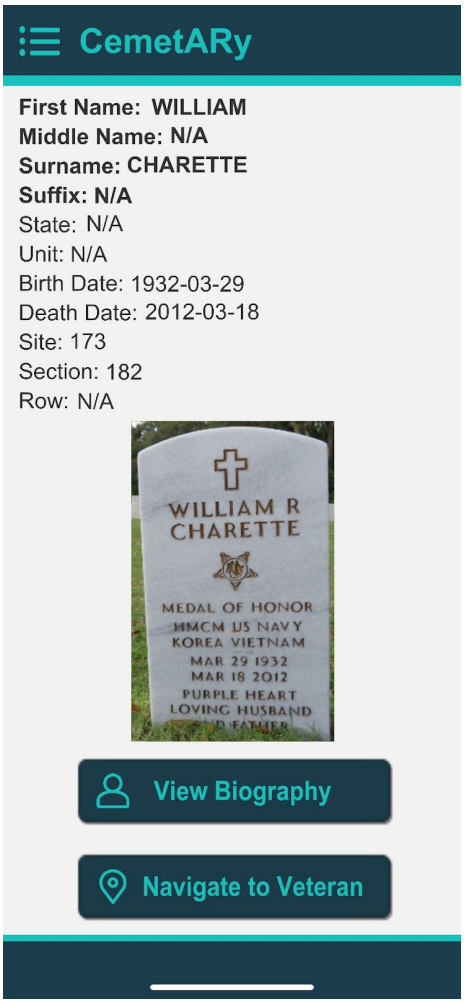

Figure 7.10 Veteran Biography Screen in Application

The Veteran Biography screen populates all of the data on that veteran stored in the database, and if a biography is available for the veteran it enables the View Biography button. If the user clicks this it will open a web url of the veteran's biography. If navigation information is available the Navigate to Veteran button will be enabled. This button opens the 2D Navigation screen, leading the user to that veteran. There is also a collapsible menu screen at the top left corner of the page, which will give them the options to: "Return to Home Page", "Find Veteran", "Scan Grave", "View Recent Veterans", and "About", as shown in Figure 7.11.

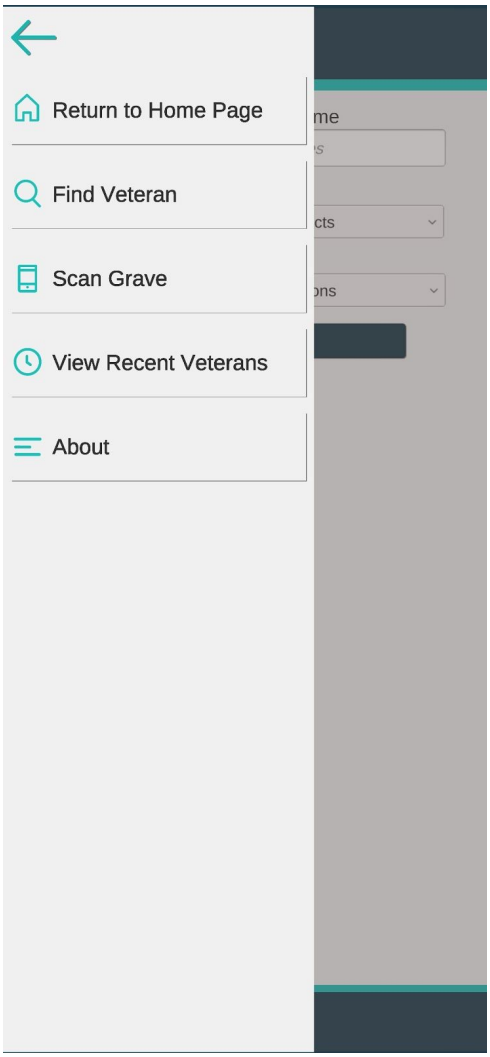

Figure 7.11. Off Screen Menu in the Application

This collapsable menu will be available from all of the different screens, except the Home Page.

The navigation screen will first begin in 2D mode, which allows the user to navigate great distances more effectively. An example of what the 2D navigation screen looks like is shown in Figure 7.12 below. This screen has a straight line connecting the user to the section the veteran they are looking for is located in. It

updates based on the user's current location and rotation, and provides map zooming and dragging to view other parts of the map.

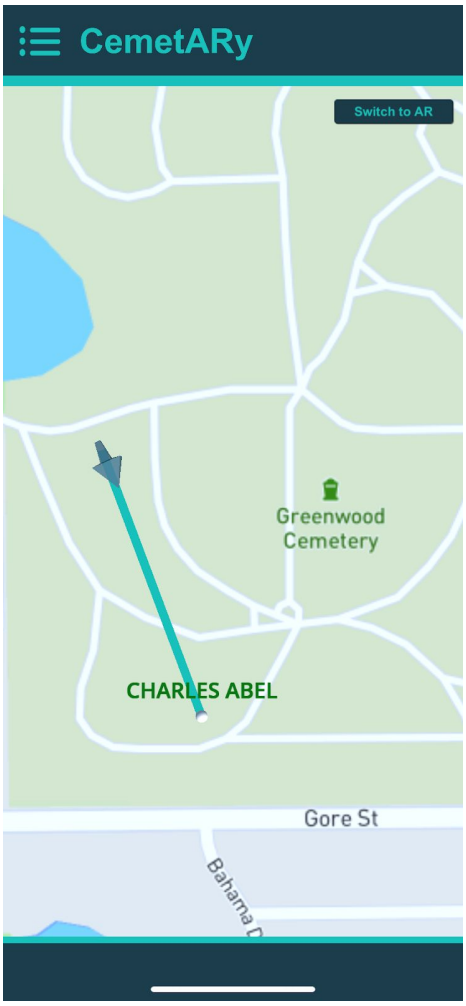

Figure 7.12. AR Navigation Screen of the Application

The 2D Navigation screen allows the user the option to switch to AR navigation. If they choose to do this then the phone's camera will be displayed on their screen and a blue line will appear from the user's phone to the veteran's location, as shown in Figure 7.13 below.

Figure 7.13 AR Navigation Screen of the Application

The blue direction line also has distance indicator text alongside it, allowing the user to easily know how far they are from the headstone. This text can be converted from feet to meters, and the user's choice will remain persistent once the app is closed down and reopened. Once the user gets within range of their destination, a 10m radius ring will appear on the screen showing the user the approximate location of the grave, as well as text that informs the user to look inside the ring for their headstone. Due to the current level of GPS technology

and accuracy, we aren't able to indicate the specific headstone that belongs to the user, but we are able to get the user in an easily searchable area.

The final option for the user is to scan a headstone to view information about the veteran. If the user clicks the Scan Grave option on the Home Page or in the Main Menu the application will switch to a screen showing the user's camera view, as shown in Figure 7.14.

Figure 7.14 Scan Grave Screen in Application

The user will be able to hold their phone up to a headstone and read the text on it by dragging crop boxes over the text. After the scan is complete the application will switch to the Search Results page, the same as with the Find Veterans option. From the Search Results page they will be able to choose the veteran they are looking for and the application will switch to a biography of them.

7.3.2 User Experience Design Flow

The diagram below, Figure 7.15, represents the user experience design flow.

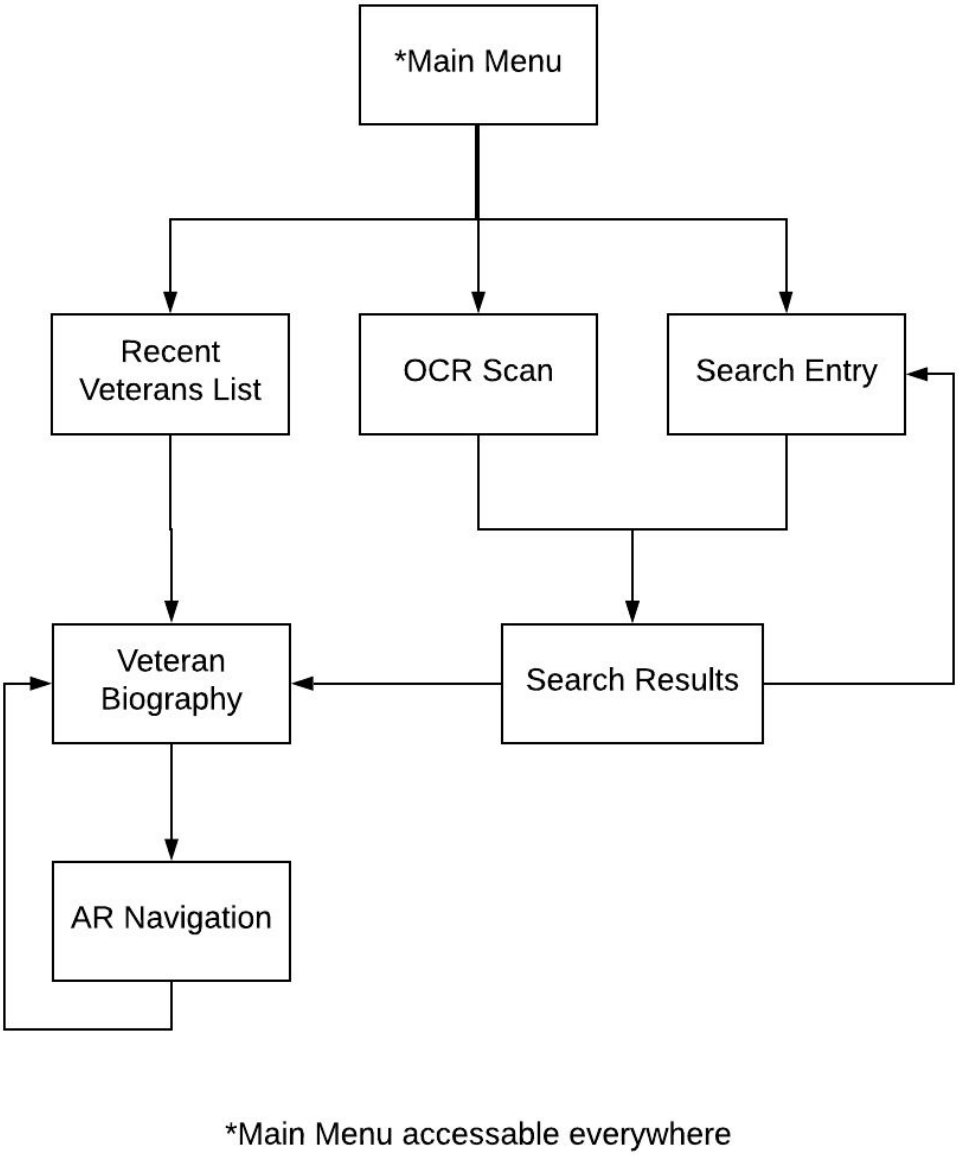

Figure 7.15 User Experience Design Flow Chart

Each of the boxes in this diagram represents a screen or page that the user will be able to view, and the direction of the arrows connecting the boxes show how each screen leads to the other when interacted with. The user will begin on the Main Menu, or Home Page, and from there will be able to navigate to all of the other pages following the flow of options indicated in the diagram above. The

Main Menu will also be accessible from all of the pages, through a collapsible menu toggle on the top corner of the screens.

7.4 Design for OCR Process

Step 1: Gather User Input

After many deliberations, for reasons that will be explained below, we decided to go with a user interface layer to assist with applying context to the input image. Fortunately, our demands of the user are low. Throughout our research all of the OCR application that we reviewed were designed to be fully autonomous. However, with a very small amount of context data we can streamline a great deal of the more problematic aspects of a fully autonomous OCR application.

The vast majority of OCR applications are designed to analyze black and white documents. Documents scanned using, not a camera, but an actual scanner. These documents, with few exceptions, are comprised of text of regular size and font. Unfortunately, headstones have none of these qualities.

Fortunately for us however, we are not using this app to process thousands of pages of old paperwork. In our expected use case, a user does not have hundreds of headstones lying around that they need to have processed. Instead, users are expected to scan a headstone for more information about the veterans once our app has guided the user to the physical location of the headstone.

Below is an image of the tutorial screens (Figure 7.16) displaying a our interface capable of gathering all the context that we need. As the image makes clear, we need not ask much of the users. Explained simply, the user is prompted to swipe or tap their finger on the camera feed to highlight, in order, the first name, last

name, and conflict.They also have the option to select the cemetery from a dropdown menu.

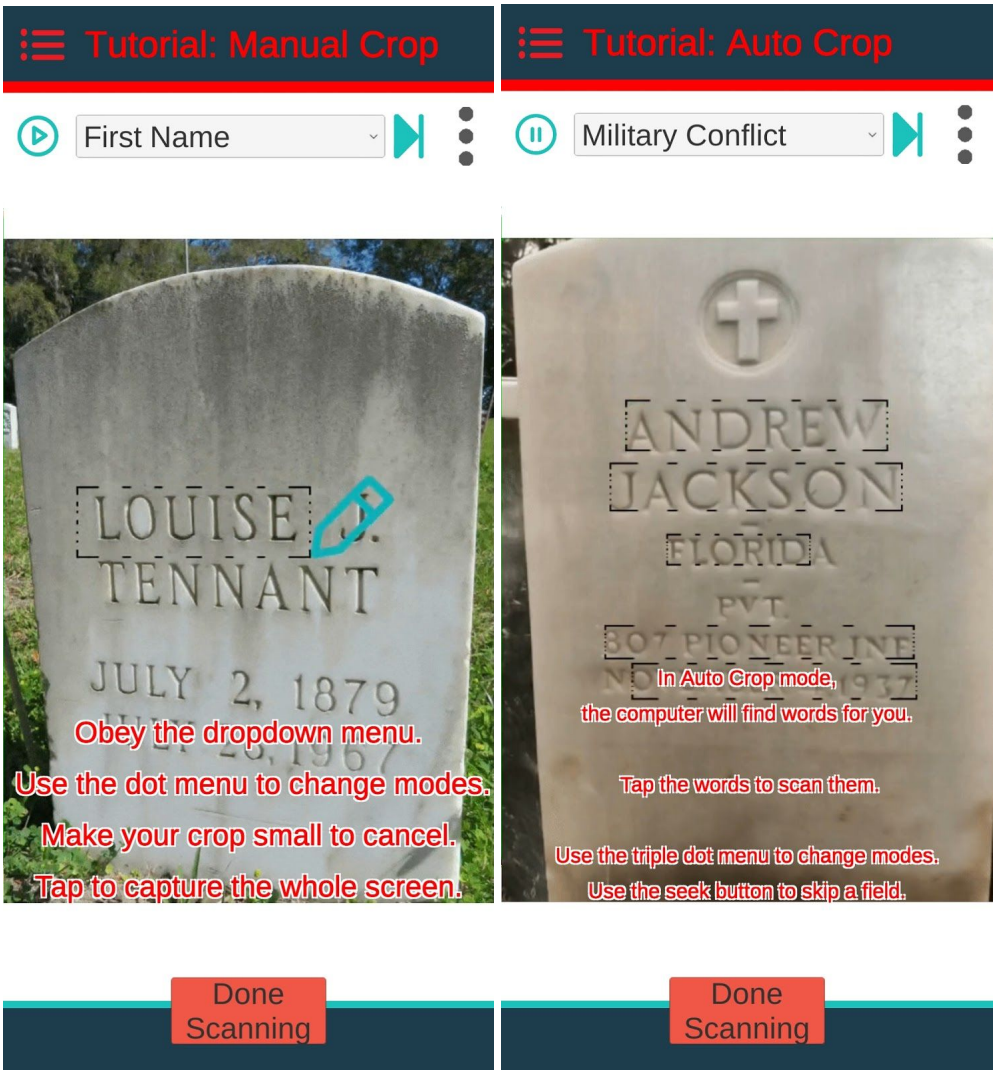

Figure 7.16 OCR UI tutorial

It is for these reasons, and more explained below, that we have a small advantage. Requiring user input would usually, in most other OCR projects, cause a substantial, bad bottleneck to performance. But we only need to process one headstone for a user at a time. And in exchange, we have a much smoother implementation process ahead of us.

Step 2: Correct Skewed/Arched Text

A skewed image is most often created when a document that is scanned at an off kilter angle. This means that when the image is rendered normally, the document is displayed in a way that a naive OCR application would find incomprehensible. In our application we will have similar issues in specific cases: relaxed phone grips, and arched text on headstones.

When resolving skew introduced by the user, it is similar enough to the normal instance of skewed scanned documents that we could employ the usual method of detecting slant. However, because we are employing user input, we can bypass a great deal of math that would otherwise slow down our app. Not that the process would be too overtly costly, but every sliver of time saved gets us closer to having a real time OCR application.

Because the user highlights the relevant data for us, we do not need to spend time processing the entire image, calculating skew etc. When the user strikes through the text they gift us with an approximate crop and because text is mostly regular, we also get an approximate skew.

After refining the bounding box to ensure that no characters are cropped in twain, we can correct the skew of linear text by rotating the cropped image for classical OCR processing.

However, the somewhat rare but still relevant presence of highly curved arches of text complicates our process. Fortunately, user input once again becomes helpful. There are two major steps to correcting for arched text: “the text string identification step that associates adjacent [connected components] into text strings and the alignment step that processes each text string individually and transforms it into a form suitable for OCR.” [22]

By gathering user data we bypass step one entirely and can focus our efforts on step two: the string transformation. Kasar & Ramakrishnan used a lot of math to detect a string and fit a B-spline to their string. For our purposes however, we can forgo most of the math and use the user data instead of a B-spline.

By detecting the characters individually using the process outlined in the OpenCV Character Detection section, we can rotate the average normal of the user's curve through each character to achieve character rotation that is more friendly to Tesseract. This approach also works for normal skewed lines of text reducing the amount of extra scenarios we need to program for.

Step 3: Segment Characters

The Tesseract api does include its own process for character segmentation. Making use of Tesseract's implementation of character segmentation would make things a lot easier for us. However, Tesseract cannot currently process curved or arched text. Therefore, we will be implementing the methodology reviewed in the OpenCV Character Detection section.

Step 4 : Analyse Characters

The main reason for using character segmentation instead of the more modern line segmentation method is that the words that we are detecting are not in a normal dictionary. They are exclusively names and dates. Characters and numbers taken separately. The advanced features of line segmentation do not help us here. In fact, by analysing the characters individually opens the door to a trick particular to our use case that works even better than the classic method of spell check error correction. More on that in Step 6 of this section.

Step 5: Apply Wildcard Threshold

Before we explain our error correction method, there is a bit of setup we need to establish. Once we put the cropped character into Tesseract, we will need to check Tesseract's confidence in its result. If Tesseract is not certain enough in its result, then we will need to replace that individual character with the wildcard: "?", an underscore in SQL.

Exactly the threshold that we will be using is to be determined in testing. If the threshold is too high then we will get too many wildcards. If the threshold is too low, it is even worse. With a low threshold we would get incorrect characters making the SQL query unhelpful.

Therefore, we intend to implement an adaptive solution that starts with a high confidence threshold. The number of wildcards would start out high and decrease as the threshold is lowered.

Step 6: Query Database for Matching Entry

As the number of wildcards in our query decrease, the number of results from the database will decrease. As we make queries of the server we will keep ahold of each result until the SQL query returns a json object that is not empty. At that point we replace the json object we were holding with the new search result. Once the SQL query returns a json object that is empty, we are ready to render the list of veterans to the user.

Step 7: Render List of Matching Veterans

Ideally, the json that the database returns right before the empty json contains only one veteran. In that case we can skip the list render and go straight into

rendering the veterans bio. If not however, the list should still be short enough for the user to easily navigate at their leisure.

8. Building and Testing

8.1 Database

8.1.1 Building the Database

MySQL Workbench was used to create the database tables and their appropriate relationships. The database is hosted by using Heroku's ClearDB service, which will host a simple SQL setup which can be dumped, which could then be hosted on a different service, such as a VM, in the future. This process should be completed by the end of May, as it is an integral part of the project.

8.1.2 Building the API

The API was initially planned to be developed over the summer, in order to provide the rest of the team with an efficient way to access the database. However, issues with setting up the database itself pushed this process back. The database was initially to be hosted on a VM, and that ended up posing serious problems about a month into development, and the system had to be switched over to Heroku. The API ended up being developed over the fall semester in an Agile methodology fashion, with all of the most important endpoints being developed first, and base functionality being covered, and then implementing the less important features as time went on.

8.1.3 Testing API responses and requests

In order to make sure the endpoints are returning the correct data at all times, the team has opted to use Postman™, an application which allows for quick and easy testing of deployed endpoints. The Express web server will be hosted on a local machine for testing, and will connect to the Heroku hosted database. Postman™ can then be used to build JSON requests in the same format which will be expected from the website and mobile application. In order to test any edge cases, the requests should account for errors associated with the OCR feature, such as missing characters, missing words, veterans who do not exist in a specific section or cemetery, and other naturally occurring issues associated with the headstones. For the navigation endpoints, the API will be sent requests containing veterans, and then checked to make sure the points returned are the correct points associated with that veteran. The last major point of testing will be to check the speed of calculating the centroid of a section on the server as opposed to the mobile application calculation speed.

8.2 Mobile Application

8.2.1 Building the Application

The application will be built in Unity. It will be built to be deployed for Android and IOS. Most of the UI for the application was developed early in the Fall semester, as well as connecting to the backend. Functionality for navigation and OCR were finished during Fall semester, following with our Milestones. The application was developed using the AR+GPS for Unity resource that our sponsor sent us. This resource accounted for the AR navigation sections of our application. We also used the Mapbox Unity plugin to develop the 2D navigation.

The navigation in the application is split into two parts. First, the user will be shown a 2D map of the cemetery to navigate to the veteran with. Then the user has the option to switch AR, which will use a distance line to direct the user to the right headstone. The API will supply the data for the headstone navigation points that the GPS navigation portion of the app will use.

8.2.2 Navigation Testing

Testing for navigation was mainly split into two parts. The first part was testing on the UCF campus with a test veteran gps point inputted in the database to ensure that the 2D and AR navigation are functional. This was tested dozens of times to ensure accuracy. The second part of testing was testing in the Greenwood Cemetery to ensure that the application worked in a cemetery. We tested both the 2D and AR navigation in the cemetery to navigate to multiple unknown headstones to test out ease of use and accuracy.

This testing was performed by multiple people on different types and versions of phones to check that the navigation works correctly for most users. The testing

occurred throughout the building process of the navigation, and was also tested at the end when the application was completed.

8.2.3 OCR Testing

Testing the OCR will mainly involve two steps. First, the OCR needs to work when we scan images of headstones. Second we need to test the OCR on headstones in the real world. Luckly, our sponsor Dr. Giroux has already supplied us with both test images as well as a real world test location at Greenwood Cemetery.

8.2.4 Connecting the App Front-End to API

Connecting the application to the API, and in turn the database, is a crucial element of the application, so this portion will involve thorough testing. Getting a basic connection of the application to the API is currently planned to be completed over the summer, but our stretch goal for the summer also involves getting most of the basic portions of the application designed and connected to the API. Testing for this will first involve a base test just to ensure that the application and API are passing information to each other.

After the OCR and navigation have begun to be implemented, and the UI for the application is designed within Unity, we plan to connect the appropriate parts of the application to the database, so testing to ensure that those specific parts are able to receive and send information to the database will be required. For navigation, this will involve ensuring that the application is able to receive GPS points for both the different sections and the specific headstones within those sections.

8.3 Website

8.3.1 React Integration Testing

As stated previously, at the beginning of the design for the website front-end of the project we started with an initial design with the use of HTML, CSS, and Bootstrap. This was done initially because of previous experience with these languages and to provide a template for how the website could potentially look like when it is fully completed. This was also important to create because of the need to provide our sponsor with progress on the design of the project and to confirm with her that we are at an understanding of what is required for the website portion and to help assure ourselves that we were on the right track. This section is used to provide insight on the process of integrating React components into the initially developed website that we started with.

8.3.2 Initial Difficulties

To start off, when implementing React into what was already coded in HTML, the first challenge was converting to the new proper syntax. For this process, it was decided that JSX would be a useful tool in helping with this transition. JSX is a syntax extension of JavaScript. It was helpful because it still allows the use of the same tag syntax that is used in HTML. Although it is not exactly the same, it is much more useful than having to rewrite all of the initial code over to a completely new language. JSX was used to turn each different section and division in the HTML document into their own separate elements. This was one of the main differences between the two languages.

Although JSX did provide the ability to continue to use the initial code that was written using tags and sections, there is still some notable syntactical differences when transitioning the code over. In HTML, it is allowed that inline tags that are

one element could be written without closing tags and it will run just fine. JSX however, does not allow this functionality and it requires that each element with an opening tag must have a corresponding closing tag. This is similar to the requirement of most programming languages that every block needs to be closed. For example, every open parenthesis must have a corresponding closing parenthesis as well as every open bracket needs a closing bracket to contain the blocks of code in order for the compiler to process information such as what variables are in what scope and so the code is executed in the proper order intended.

This was more of an issue than expected when beginning to test code that was previously written in HTML that included statements that didn't have closing tags for convenience and because it was not required. This is where most of the errors arose from when initially starting to work with React and JSX. This was not so much a difficult challenge, but more of a tedious task having to go through all of the lines of the existing code and making sure that every single opening tag had its corresponding closing tag.

Other initial errors included some syntax differences between HTML and JSX. This was less troublesome because there were only small differences such as instead of using the keyword “class” to define a class, JSX uses the keyword ”className”. So instead of needing to inspect the code line by line, all that was needed was to utilize the find and replace functionality that Notepad++ provides in order to replace all instances of the word class with the word className.

Although, when researching a useful way to debug JavaScript code, Google Chrome's developer tools seemed to be a great resource to utilize. Google Chrome's developer tools proved to be useful for helping with debugging the code by providing useful information on the errors that were occurring and

helping narrow down what lines of code were causing these errors. At first it was not clear as to what was causing these errors and Chrome's developer tools debugging features really helped explain the errors and why they were occurring.

These tools allow the ability to set breakpoints within the code so that it can pause during execution and when an error occurred, we could then step through the code, one line at a time, to aid in the investigation of what code is causing errors. It is a tool that we definitely continued to use when testing new development on the front-end of the website.

8.3.3 Connecting Web Front-end to API

Without a connection to the back-end database, the front-end website would just be a nice looking but completely useless user interface. The website needed to have access to the API in order to input new information into the database and also to verify user login credentials. The first endpoint that we used to start testing successful connection to the API was the endpoint of creating a new user on the website.

The goal was to gather the information that the user enters into the create new account form that was created for the website. Once the user chooses to submit the data, we then send that data over to the API in json format. We did this first because we needed to test the data being sent to the API and make sure it is being retrieved properly. The database then needed to retrieve that data from the API and store it into the corresponding table that will store all user information. This was the initial API testing to make sure that the website can successfully send over and store data into the database.

Once storing new user login credentials was complete, we then tested verifying those credentials when users attempt to login to their accounts. In order to test

this, we had a few accounts that have previously been created and currently exist within the database. Then, we tested login attempts on those different accounts. The reason for testing multiple accounts is so we could ensure that a user account can be found through a list of existing accounts. We then tested comparing the entered user credentials with what is currently stored within the database at the time of the login attempt. When we were successful in finding a match to both the username and password located within the same record, login is approved and access to the website is permitted. This also checks for the event of invalid logins when the entered information does not exist within the database. A message is shown to the user that the login was unsuccessful for this reason and they will have to attempt the login again.

After successfully creating new user accounts and testing login functionality, we were then able to move to the next step, which would be allowing users to enter their cemetery information into the database. We first had to test creating a new cemetery in the database, we have been using the Greenwood Cemetery that we are building this project on as a test case. We entered all of the relevant information pertaining to that specific cemetery including the city and the name of the cemetery. Once a cemetery is successfully submitted, its is given its own unique ID that is used to enter the other information for that cemetery. We tested sending this information to the database through the API, making sure that it is entered into the cemetery table and that all of the required fields within that table have been filled in with the correct information.

The next step was entering information on the veterans that are buried within the cemetery. We tested to make sure that all of this information is stored properly, and that the information entered will be specific to that cemetery. This helps narrow down searches when the mobile application attempts to find a veteran within the database. Finally, we needed to enter the GPS coordinates for the

cemetery so that the mobile application could use that data for the GPS navigation within the cemetery.

9. Project Summary

This project summary details the choices of the team as far as design, implementation, and testing. As in any development project, these choices may shrink, change, or grow depending on the discoveries of the team during testing.

On the back-end, the project will use MySQL as the driver for the database, which will be hosted on Heroku for building and testing, and then exported and imported into a virtual machine as it becomes available to the team. The web server containing the list of API endpoints will be written in the Javascript frameworks Node.JS and Express. The endpoints will be organized in the most up-to-date manner, using separate files and routing. To ensure the security of these endpoints, the team will use JWT tokens, and a Javascript library Passport to add authentication. Passport is added to each of the endpoints to check if a valid user has logged into the site, creating one of these tokens. The tokens are not necessary for endpoints related to the mobile application, as the endpoints do not affect the database, and are publicly accessible information. For testing the database setup, queries can be executed directly in MySQL Workbench or PHPMyAdmin, and endpoints will be tested using Postman™

The website front-end provides functionality for administrative users that wish to enter the information on their cemeteries within our database. The React framework was used to create the fully responsive website. Users are able to create a new account if they have been provided an access code and they will then be allowed access into the website. An administrator from a new cemetery

will be able to add their cemetery to the system and existing users will have the ability to update information on cemeteries already existing within the database. Once a cemetery has been added, the user can then input all of the information that will be stored for use in the mobile application portion of this project. This important information includes the GPS coordinates of each individual veteran section of the cemetery so that the mobile application can utilize that to help navigate users to those different sections. The other information entered will include detailed information on each individual veteran that is buried in the cemetery as well as the GPS coordinates of their respective headstones. This is used in the mobile application as well to help the people visiting the cemeteries by navigating them to the individual headstones and providing them the information about each of the veterans within the cemeteries.

The application front-end is for the users visiting the cemetery. The user interface for the application was designed using Unity, and the AR and GPS portions of it was implemented using the AR+GPS for Unity asset provided to us by the sponsor and the Mapbox plugin. In the application the user has a choice to search for a veteran to find their biography and then navigate to their gravestone, or to perform an OCR scan on a headstone they are already at to view the veteran biography. The navigation portion of the app allows the user to switch between 2D and AR navigation to effectively find a veteran's headstone.

The OCR mode of our app has seven main parts:

1. Gathering User Input
2. Correcting Skewed and Arched Text
3. Segmenting and Isolating the Characters
4. Analysing and Processing the Characters in Tesseract
5. Applying a Wildcard Threshold
6. Querying the Database for Matching Veteran Information

7. Rendering a List of Matching Veteran Information

Once the user engages the OCR mode of our app, they will be prompted to highlight the information that they want searched by swiping their screen. Then we will use a combination of OpenCV and Tesseract to transform the camera data into an OCR readable format.

The transformation stage of the OCR process needs to account for the arched typesetting present on some of the headstones.This is covered in greater detail in the Preliminary Design for OCR Process section (7.4). Suffice to say that our proposed solution leverages the unique circumstance and use case that our project involves.

Once the image has been parsed it will be used to query the database for the relevant veteran information. If there is more than one search result then the user will be shown a list of matching veterans. If there is only a single match, then in order to streamline the user experience, the user will be directly shown the veteran's biography.

The ideal performance of the OCR mode of our application will have the user be shown directly to the correct veteran's biography. However, we do not want to be too aggressive about getting a single result. Accuracy must be accounted for as well.

For this reason we will allow the user to make the final call in the circumstance that we are less than one-hundred percent certain in the accuracy of our algorithm. Our users are not looking to wiz through dozens of search results. Our users want the information of the single headstone in front of them. The

acceptable margin of error is directly tied to the acceptable margin of error for user frustration.

Relatedly, in an effort to curb any potential user frustration, users will have easy access to their previous searches. The users will be able to see veterans they had previously viewed the biographies of in the Recent screen and again view their biographies if desired.

10. Closing Thoughts

Following are a short section of closing thoughts by the team. Successes, failures, and things the team as learned over the year are included.

Joshua Lecas

This project allowed me to really sink my teeth into database and backend development. I had not used MySQL in any applicable way, and true database design and management was something I had never worked with aside from one database class that I had taken. That being said, I have some closing thoughts about the database itself. Currently, the veterans in the database do not have any conflicts associated with them, or branches, race, etc. This is due to the issues that I faced when creating and importing the data provided to us in the initial stages of development. I was not able to figure out the process for cleanly adding this information to the database without causing drastic changes to the database itself, potentially breaking the project. In the future this information must be added in manually. Another point of failure is not having endpoints for manipulating data in the secondary tables such as Conflict and Branch. This was noticed late into the project and there was not enough time to easily implement this. It may not be a massive issue, as this data can be imported into the database rather easily using MySQL or some other database management tool, and this data is not going to be manipulated often, as new conflicts are not going to be added more than a dozen times.

Another issue I ran into was the format of the data that I was provided. The table of veterans included the conflicts of the veterans, but the conflicts of the veterans were not in a consistent format as the database was requested to follow. For example, the purpose of having a many to many relationship was to have a single entry for each conflict inside of the conflicts table, and a third table would

hold the conflicts that a veteran had been a part of. However, in the data provided, the veterans had conflicts listed as “WWI & WW2”, and when importing this data, there were entries in conflict for WWI, WW2, as well as “WW1 & WW2.” This was an issue noticed late into the project, and required much more effort to fix than was necessary for project completion.

I had a lot of successes with this project as well. I successfully set up a professional project structure for the web server, with separate folders for different endpoint files, and formatted my code in these files which is easy to understand for a new developer to come in and add or change features. I learned many professional coding standard for encryption, authorization, and security which I implemented throughout the code, making our project an impressive back-end supported application, considering the short amount of time we had for development.

Jonathon Towne

A lot has happened over the last two semesters. I have learned a great deal about OCR, Unity, mobile cameras, color channels, POST api calls, and general C sharp coding. Most of what I learned about Unity was how much it struggles to cooperate with anything outside of its ecosystem.

Initially I had planned to get Tesseract, the OCR program, into unity. However, due to many compatibility conflicts, after several weeks of importing many dependencies I began to run out of time. The crux of the issue was that Tesseract relies heavily on a great deal of external libraries. On a Windows platform, there are two libraries that need to be imported. However, in order to develop for mobile platforms, a great deal of Windows libraries needs to be imported. This did not stop me from trying. Unfortunately, Tesseract requires .Net v4 and Unity only supports .Net v2. I spent some time trying to trick Unity into

running .Net v4 (with some success). In the end though, there is only so much time I could justify spending on this without a guarantee that Unity would still even run afterwards.

Unity's default ability to use mobile cameras is limited to reading the raw camera data. However, the textures in mobile devices are rotated (and inverted on iOS) on account of the hardware being installed with a physical ninety degree rotation. Unity does not have a standard way to resolve this. It also has no way to rotate the object that the texture is rendered to, without also rotating the objects' coordinate system. This is a big problem for my project, because detecting a press on a rotated coordinate system greatly complicates the process of calculating the corresponding pixel coordinate data. For this reason, a sizeable portion of the work I did this second semester was coding texture functionality into Unity.

It was not always a struggle however, once I got to grips with Unity's wack process things progressed much more smoothly (It is not actually all that crazy, it is just meant for game design, not for mobile apps and texture work). Google Vision is an incredible feat of machine learning and their POST api calls made up for the time lost struggling with Tesseract.

After I got the base infrastructure coded, I developed the autoCrop mode. Because of the limited time remaining before CECS and my previous experience fighting Tesseract, I elected to purchase the asset version of openCV. The company I purchased the asset from created an effective interface between Unity's C sharp and openCV's C++. However, this meant that I could not use most of the openCV documentation available. The documentation that a certain company provided was terrible and suggested inefficient and straight up broken code (They used functions that required a Mat of a very specific type on one line

and on the very next line used the same Mat with a different function that required a different Mat of a very specific type). Once I debugged their code however, the interface worked fine. Unity did not cooperate however, it was revenge of Unitys coordinate system remixed again. Basically, Unity has a bunch of functions that allegedly allow someone to convert between their many coordinate systems (screen space, GUI space, local space, world space, etc. not pixel coords though). I say allegedly because many of their functions would take in a point and return the same exact point. In the end, I just wrote the conversion functions myself and everything worked out great.

Katie Beason

This project allowed me to expand my knowledge about developing applications and gave me valuable experience in working within a team. While I came into the project with knowledge on how to use Unity to develop 3D applications, I had never attempted to make a phone app in Unity or develop AR. This project allowed my knowledge of how both of these aspects of development work grow extensively and will be valuable information for me to have going into future jobs, or even for personal development projects.

I was concerned at first about developing the application in Unity since I was the only one on the team with much experience in it and I had never even developed a mobile application in it, and while there were some setbacks and frustrations with user interface elements I ended up being glad our sponsor wanted us to develop in Unity since I was able to use my familiarity with it to more easily accomplish some part of the app. The user interface probably presented one of the most persistent challenges for me while we developed the application. We were able to get a rough UI up fairly quickly, but as we made improvements to it and attempted to test it on different devices we kept finding we had a lot of sizing issues of different app elements since Unity doesn’t handle UI dynamic sizing

very well. We have since tested it on many different phone sizes though so this hopefully should be an issue.

The AR also presented a challenge since I had no experience at all in that, and we struggled for a while with coming up with an idea of how to represent the GPS uncertainty of where the headstone is in Unity. I am happy with how the final result turned out though, when I tested it myself in the Greenwood cemetery I found it was intuitive and easy to use.

I also learned a lot about connecting Unity to an API, converting Unity code to JSON, and having persistent data in a Unity application. Before this project, I had no idea how to do any of that, but those are now useful skills I have that I can bring into my future job.

Chris Monteverde

This project has helped me expand my knowledge and experience in web development tremendously. Before this project, I had taken a couple of web design and graphics courses. I had learned the basics of HTML, CSS and JavaScript which was definitely useful when developing this project. However, this project required that we use a more current framework such as React or Angular. We chose to go with React and I can honestly say that I am satisfied with the decision. React is one of the most popular frameworks used today, so gaining experience with it was definitely valuable for me. Because of its popularity, there is an abundance of documentation and resources available that helped aid me in implementing many of Reacts features during the development process.

I also gained experience in connecting my website front end to the database backend through the use of an API. Before this project, I had little experience

with this so gaining this knowledge while creating this project was valuable. Some of this new knowledge included the use of JSON Web Tokens when making requests to read or write from the database. This helps keep the database secure because each administrative user is given a unique token from the API when they log in that is stored in Reacts “localStorage” variable and is then used every time they send a request to the API that makes changes to the database. This is also much simpler than requesting that the user enters their login credentials each and every time they wish to send a request to the API.

In closing, I am pleased with the decision of using the React framework for the website as well as the experience I have gained while developing this Senior Design project. I have no doubt that the knowledge I have gained during both semesters of Senior Design will prove to be invaluable for my future endeavors as a computer scientist. Having to do my own research, learning new technologies, and documenting this project are useful skills that I can now bring to my future career as a computer scientist.

References

[1] Celada, Jan. “Unity 3D Not Just Games.” Medium, Medium, 28 Feb. 2015, medium.com/@raquezha/unity-3d-not-just-games-aef911e3314c.

[2] “ARCore Overview.” Google, Google, 28 Feb. 2019, developers.google.com/ar/discover/.

[3] “ARCore.” Unity, Google, 2019, unity3d.com/partners/google/arcore.

[4] “Unity-ARKit-Plugin.” Bitbucket, 7 Apr. 2019, bitbucket.org/Unity-Technologies/unity-arkit-plugin.

[5] “Partners - Vuforia.” Unity, Unity Technologies, 2019, unity3d.com/partners/vuforia.

[6] Fortes, Daniel. “AR + GPS Location Documentation.” AR + GPS Location, 2018, docs.unity-ar-gps-location.com/api/index.html.

[7] Fortes, Daniel. “Unity AR+GPS Location.” GitBook, Gitbook, 2018, docs.unity-ar-gps-location.com/.

[8] “Node.js vs Python Comparison: Which Solution to Choose for Your Next Project?” Netguru , Netguru, 30 Jan. 2018, www.netguru.com/blog/node.js-vs-python-comparison-which-solution-to-choose-for-your-next-project.

[9] “Garmin”,https://www.amazon.com/Garmin-GPSMAP-High-Sensitivity-GLONASS-Receiver/dp/B00HWL9BQ4, 2019

[10] Kurtaev, Dmitry. "OpenCV library." OpenCV library. 2019. 15 Apr. 2019 <https://opencv.org/>.

[11] Hemken, Soren. "OpenCV & Tesseract: Android Computer Vision for Dummies – CraftCode Crew." CraftCode Crew. 03 Feb. 2019. 22 Apr. 2019 <http://craftcodecrew.com/opencv-tesseract-android-computer-vision-for-dummies/>

[12] Rmtheis. "Rmtheis/tess-two." GitHub. 13 Jan. 2019. 15 Apr. 2019 <https://github.com/rmtheis/tess-two>.

[13] Megapro17, Tesseract-Ocr. "Tesseract-ocr/tesseract." GitHub. 13 Nov. 2018. 15 Apr. 2019 <https://github.com/tesseract-ocr/tesseract/wiki/ReadMe>.

[14] Zdenop, Tesseract-Ocr. "Tesseract-ocr/tesseract." GitHub. 1 Feb. 2019. 15 Apr. 2019 <https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality>.

[15] Shreeshrii, Tesseract-Ocr. "Tesseract-ocr/tesseract." GitHub. 30 May 2019. 15 Apr. 2019 <https://github.com/tesseract-ocr/tesseract/wiki/FAQ>.

[16] Mordvintsev, Alexander, and Abid K. Revision. "Image Thresholding." OpenCV. 2013. 15 Apr. 2019 <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html>.

[17] Vanderkam, Dan. "Finding blocks of text in an image using Python, OpenCV and numpy." Finding blocks of text in an image using Python, OpenCV and numpy. 7 Jan. 2015. 15 Apr. 2019, <http://www.danvk.org/2015/01/07/finding-blocks-of-text-in-an-image-using-python-opencv-and-numpy.html>.

[18] Insights, Cuelogic. “What Are The Differences Between Angular And React?” Cuelogic Technologies Pvt. Ltd., 8 Apr. 2019, www.cuelogic.com/blog/what-are-the-differences-between-angular-and-react .

[19] Willoughby , John. The Top 5 Benefits of React, 6 Dec. 2017, www.telerik.com/blogs/5-benefits-of-reactjs-to-brighten-a-cloudy-day.

[20] Srivastava, Shrikant. “React vs Angular: What to Choose for Your App?” Appinventiv Official Blog for Mobile App Development, Appinventiv Technology, 26 Mar. 2019, appinventiv.com/blog/react-vs-angular.

[21] Hale, Coda. “How To Safely Store A Password.” *How To Safely Store A Password*, 2018, codahale.com/how-to-safely-store-a-password/.

[22] Kasar, Thotreingam & Ramakrishnan, A.G.. (2013). Alignment of Curved Text Strings for Enhanced OCR Readability. International Journal of Computer Vision and Signal Processing. 3.

Appendix

A. Copyright Permissions

Amy Giroux <Amy.Giroux@ucf.edu>
Thu 2/28/2019 4:15 PM

Show all 4 recipients

To: Joshua Lecas; SmilesBlue <jontowne11@gm... ...

Jonathon Towne, Joshua Lecas, Katie Beason, and Christian Monteverde, of Senior Design Spring 2019-Fall 2019 Team 6, have my permission to use the 386 headstone photos and aerial view of Greenwood Cemetery that I took for their project.

Amy Giroux

B. Software Used

- LucidChart: https://www.lucidchart.com/
- Postman: https://www.getpostman.com/
- Notepad++: https://notepad-plus-plus.org/
- Unity: https://unity.com/
- Mapbox: https://www.mapbox.com/