Bridging the Gap Between Digital Infrastructure and Interpretive Information in Public Spaces

Student Group Six

Braedon Watkins, Edelis Molina Rios, Gian Alvarez Rujel, Robert “Tye” Riley, Omar Mosa

UCF CHDR Sponsors

Dr. Amy Giroux, Evan Wallace

UCF Mentor

Dr. Rick Leinecker

Summer 2022

Table of Contents

1 Executive Summary

The Exhibit Marker Uniform Resource Router (EMURR) is a technological instrument designed to bridge the gap between digital and physical interpretive sites originally envisioned by the Center of Humanities and Digital Research at UCF. Inherent to bridging this gap our team intends to provide a superior learning experience for students visiting physical interpretive sites such as cemeteries, museums, science centers, and art exhibits. In order to achieve this, our goal is to deliver on our designed software and hardware layers that support one another.

The web builder is the fundamental basis of the software layer we have designed. The web builder is, at its core, a What You See Is What You Get (WYSIWYG) editor. The editor may contain text, images, and video which will then be compiled to HTML pages through a tool called TinyMCE. Users authenticated as editors will be able to manipulate pages. Users that have purchased an EMURR unit can download edited pages as tour sites and host them locally in tour packs. Users authenticated as super-users will have control over roles and other elevated actions to help manage content on the platform. This is all designed in Figma, tested by Jest and Storybook, constructed with Next JS, and stores data through Prisma in a MySQL database. The hardware layer consists of a Raspberry Pi, TP Link AC1200 Router, and NFC tags. The Raspberry Pi will operate with NGINX to act as a web server, while the TP Link AC1200 port forwards traffic to the Pi at a static IP address, and finally the NFC tags will be initialized to redirect to sites on the Pi when an NFC enabled device reads them.

We have also outlined some stretch goals which will enrich the experience of EMURR but are not necessary to deliver on the team’s vision. Diagnostic stretch goals exist such as heat mapping and user comments. This way teachers can see what sites students gravitate toward and their thoughts on each of them. Teachers will be able to react to this information and edit tours as they please to improve experience. We have also outlined a stretch goal for scraping web pages such as Wikipedia or Veteran’s Legacy Project (VLP) for existing content and citing it appropriately.

With any tool including software it is important to consider legal, ethical, and privacy concerns. Being that EMURR relies on a web application with user credentials we want to ensure that all such information is secure and handled professionally. We have explored several vectors of attack and are prepared to defend against them, and other such exploits should they present themselves. Furthermore, we understand allowing users to generate content opens concerns regarding copyright. We have explored several strategies to manage this such as internalizing data and terms of service.

2 Project Overview

The design for the EMURR was originally presented by Ph.D. student Evan Wallace and Dr. Amy Giroux based in UCF's CHDR lab. Since then, the team's design has expanded on the base idea as a byproduct of considering methodology and implementation.

2.1. Description

2.1.1 Hardware

For EMURR to deliver on its intended design, the unit will provide software on an affordable and readily available hardware layer. This hardware layer's deliverables consist of lockable NFC tags, a microcontroller, and a router.

Each password-locked NFC tag will provide owners of an **EMURR unit**[1] the opportunity to refer physical sites to the digital site by pointing to its corresponding IP address. NFC tags will be initialized, before being sent out to the owner, when the unit is created. Initialization means the NFC tag will store information to point to a static IP address on its corresponding Raspberry Pi and subsequently locked from being edited after. Several 3D printed housings for NFC tags have been designed in order to stay within compliance with National Cemetery Administration (NCA) guidelines. Two popular options we have settled on using are a coin shaped enclosures and a plant stake shaped enclosure. Once initialized and housed, owners of the unit can then lock the NFC with a password of their choosing. Finally, anyone with an NFC enabled device can tap the desired tag to view the digital site and the information therein. The sites viewed in this way are referred to as **tour sites**.

[1] EMURR unit denotes all the hardware components of a minimum viable product (Pi, router, battery, set of exhibit markers, and accessories).

The Raspberry Pi will be responsible for hosting a collection of tour sites. For any sites made to be dynamic the Pi will also have to handle client interaction and API requests (See Comments, Dynamic Tour Sites, Heat Mapping in the Goals and Objectives section for more on necessary API requests). Sites may be selected for hosting on the microcontroller in a collection referred to as **tour packs**. The proof-of-concept Raspberry Pi models selected for testing are the Raspberry Pi Zero 2 W and Raspberry Pi 3 Model B+. If either present limitation in testing, the team is prepared to upgrade according to our further research.

Figure 1: EMURR unit attached to a backpack

The router will be responsible for providing a closed network service in an effective 600-foot diameter around the tour guide. In this manner, spaces such as cemeteries may have cheap and effective digital resources regardless of preliminary infrastructure at the site. The router will be carried by tour guides attached to a backpack or in some equivalent manner as shown in the figure below. The router the team selected for testing is the TP Link AC1200.

2.1.2 Software

With an effective hardware layer to lay the foundation, EMURR's software layer is positioned for success in delivering on its goals. The software layer revolves around the web application referred to as the web builder. The web builder bears the responsibility of empowering users to interact tour sites in a user-friendly and accessible interface. To be more specific, interaction with tour sites is determined by the permissions of the role that a user is authenticated to. User authentication consists of typical operations such as login, registration, account recovery, and account deletion. User permissions consist of editor, admin, and super-user.

Our initial design allows anyone on the web to create an account to become an editor and upon authentication will have permissions to edit tour sites, publish their tour sites, and view other users' published tour sites. Editors will be able to see tour sites they've edited and generate new sites. When they are ready to make these sites public, they can publish them for everyone to see. And of course, editors can see other editors' sites that have been published.

We explored two stretch goals to raise the quality of tour sites. First, we considered giving editors the ability to vote as a means to self-moderate the platform. Similarly, in light of concerns of copyright and security we may require authorization by super-users to publish, meaning all public content is filtered by trusted users. Another stretch goal to ease the burden on editors initially populating the database is to write an automated script that pulls other cemetery information, such as Veteran Legacy Project (VLP), and stores them as tour sites in the database. However, concerns over copyright and security are severe enough to consider the possibility of limiting the scope of permissions for editors. Since editors can inject arbitrary media and HTML pages to tour sites this introduces the capacity to push copywritten media or insecure code to our servers. Considering this we are prepared to deprecate the editor permissions to only be available to administrators, meaning tour sites are only visible to those with EMURR units.

Only users with EMURR units will have administrator permissions, which is a superset of editor permissions. Upon authentication an administrator will be able to load local or published tour sites to the EMURR unit into a collection referred to as a **tour pack**.

Only those affiliated with EMURR and CHDR will have access to super-user permissions, the set of all permissions. Upon authentication super-users will be able to manage user permissions, elevating roles, and even deleting content on the server. Should we meet our stretch goals they will also be able to approve sites awaiting publication.

Unauthenticated users, otherwise referred to as viewers, will not have access to the web builder but will instead be able to view tour sites hosted by the Raspberry Pi. These users with NFC enabled devices will tap an NFC tag which will point them to the tour site in its respective tour pack. This site will be read only for these users at all times. One stretch goal we have for the tour sites is to develop heat maps of tours over time by tracking MAC addresses.

The software we will use to achieve this user relationship consists of a Headless Raspbian (Linux) operating system running on the Raspberry Pi as well as Next JS and Prisma. The team will utilize testing software such as Jest and Storybook.

2.2. Broader Impacts

EMURR is a lightweight, low-cost, and portable framework that intends to improve public engagement and education at interpretive sites for students.

The first way this will be achieved is by providing a more accessible experience for students. With an effective router Wi-Fi diameter of 600 feet, students are free to explore exhibits without being forced to follow any particular path nor pace. This has the potential to unburden the experience for all students, including those with disabilities such as attention deficit disorders or paraplegia.

The second way in which lowering costs for nonprofit organizations otherwise referred to as interpretive sites. Addendums and edits of already formed exhibits can be tedious, time consuming, and costly. This process often requires bureaucratic tasks in the form of write ups for proposals, grants, and funding. While larger interpretive sites can secure said funding, many smaller facilities cannot. With EMURR these sites can be modified, maintained, and accessed digitally therefore lowering costs and time spent. This will ultimately raise the diversity of interpretive sites that can thrive by supporting their needs.

2.3. Personal Motivations

The following section contains a brief statement of motivation for each of our five team members describing the personal interest and motivation to select EMURR as the Senior Design project.

2.3.1 Braedon Watkins

Ever since I was young, I had a great interest in many technical subjects, not least of which was programming. For years I spent my time after school contributing to a local Boys & Girls Club robotics team. I still remember pleading with the lead programmer to teach me everything they knew. I have since spent a lot of time involved in all too many programming related subjects, some of which utilizing microcontrollers like the Raspberry Pi. In conjunction with an interest in emerging applications of NFC technology I knew I had to work on EMURR. I am honored and excited to be working on this project. I can't wait to see what we can do together!

2.3.2 Edelis Molina Rios

My motivation for selecting EMURR as my Senior Design project is to leave an impact on the underfunded organizations by designing a cheap and readily available software and hardware infrastructure to access digital information in areas with a lack of Wi-Fi and cell phone coverage. I had not yet had the opportunity to work on a project that has a meaningful impact on the community, specifically, a project to help small groups and organizations to overcome economic barriers to maintain digital interpretive information.

Another reason for choosing EMURR was the use case Dr. Amy Giroux selected to pitch the project idea. She mentioned that UCF history students had written over 200 biographies for veterans buried in three cemeteries here in Florida and two cemeteries in France and while these biographies are available on a website for public access and there exists a k-12 curriculum that references them, accessing the biographies on the ground in the cemeteries is cumbersome. I think this project will allow teachers with no programming background to create tour packages with a few clicks and take their digital access with them to make the experience on the ground more engaging for k-12 students.

Additionally, I have worked on several projects throughout my undergraduate career at UCF; however, all of them have been mostly software related. With EMURR, I will also have an opportunity to interact with a Raspberry Pi and a router to design the hardware infrastructure. I am looking forward to working with the team and our sponsor on this project to build a product that is intuitive to use by non-programmers to create digital content and scalable to any location lacking a reliable internet connection.

2.3.3 Gian Alvarez Rujel

My motivation for this project stems from my initial discovery of programming at the beginning of my college career. Interested by the deep range of problems one could solve with coding, as well as the intricate projects people could create, I have spent countless hours learning how to develop applications, as well as the many uses of various programming languages. An example would be that I have put a lot of effort into designing and developing sample websites as personal practice in my spare time away from coding assignments or other academic based projects.

As EMURR is my first real technical project, I am excited by the potential for change its applications can have in both classrooms, historical sites, and even museums. More so, discovering alternatives for API development and NFC tag uses are topics I am already familiar in and am looking forward to further develop. Aside from that, the fact that this project will allow me to delve into areas that I have little familiarity with, such as working with Raspberry Pi's or front-end development, is another aspect that has made this project thrilling to work on.

2.3.4 Robert "Tye" Riley

Since the start of my computer science degree, I've been more intrigued with how we can make our computers an extension of our bodies. Most of my work in research revolves around microcontrollers and how we can incorporate mechanical systems to help the body move and function on an everyday basis. I gained knowledge in the power of single board processors and how they might be useful to process specific sensors and singular scripts. So naturally, I began to theorize the power of portable computing and how it incorporates with our cellular devices. Oftentimes, phones lack the physical hardware and resources to supply a sustainable internet connection and reliable extensibility than its desktop counterpart. After seeing this project, I was inspired to see how we may incorporate this new systematic technology to make the internet and processing power more portable and accessible to the world.

2.3.5 Omar Mosa

My journey of pursuing the field of computer science started four years ago, but I've always had an interest in computers since my early childhood. Before I knew I would pursue this path, I wrote various kinds of scripts and learned web design in high school. I started programming short term and long-term applications during college outside of coursework, which allowed me to learn new skills and build upon existing ones. I take great pride in my work and always strive to produce high quality work I can be proud of.

This will be my first project working with NFC tags and a Raspberry Pi. I'm excited to work on this project to learn new skills, and to have a real-world impact for teachers and students. I am most looking forward to building a product with an excellent user experience, and I believe my experience with real world applications will help our team accomplish that.

2.4. Legal, Ethical, & Privacy Concerns

2.4.1 Security

As with any application that allows users to make sensitive changes, security is paramount. We need to ensure digital access is secure to prevent malicious users from editing pages or giving privileges to other malicious users. Security is important not only for authentication purposes, but also for safeguarding private or personal data such as emails and passwords from attackers. While it is important for users to practice good password security on all websites, we cannot simply assume that this will be the case. People reuse email and password combinations all the time, so a security breach where passwords are stored improperly on our application can have catastrophic results for our users.

The following is an exploration into some common security concerns we are aware of and are committed to defending against.

2.4.1.1 Cross-Site Scripting (XSS)

Cross-Site Scripting, abbreviated as XSS, is one of the most prevalent vulnerabilities on the web. In point of fact, it was only just a few years ago in 2019 when Masato Kinugawa, a bug bounty hunter, discovered an XSS exploit in the main page of Google. If Google was not safe from XSS then we figure it is worthwhile to investigate if we are. Not only is the exploit pervasive but it is severe. XSS allows malicious actors to execute arbitrary code, usually JavaScript, on the websites that have introduced such vulnerabilities. The vulnerability to XSS is most generally introduced when a website allows users to provide input which can then edit the page on re-rendering. For the purposes of consistency among examples we discuss only instances containing HTML with JavaScript but we understand XSS can also affect XML and SVG alongside any scripting language. More specific methods of XSS we will go on to investigate break down to reflected, stored, DOM, and mutation XSS.

Reflected XSS is possible when a server returns user input directly into HTML. Suppose as a simple example we have a header <h1> tag which will receive a variable determined by user input submitted in a text box. If in our input we enter `<script>alert('iamhaxxer');</script>` then the HTML reflected back to us will have a header containing our input. The script tag will be read on re-render and sure enough our script is now being executed as if it were intentionally written on the site. This may not seem like a meaningful attack if this only occurs locally but for any GET request which shows in the URL an attacker can then format this as a link. What happens next can range from phishing emails to social engineering to get someone to navigate to the site. Whoever unfortunate enough to go to such a site will be victim to whatever arbitrary code an attacker has constructed

Stored XSS is like reflected XSS but more surreptitious and extensive. Instead of reflecting arbitrary code back to the browser the attacker now wishes to store this code in the web application server. Doing this means that anyone who visits the site will be affected as opposed to just the people that the attacker personally targets. The possibility for stored XSS opens up when sites include elements where a user's input becomes stored information, a prime example being comment sections. If your comment were to include script tags in a similar manner to reflected XSS `<script>alert('haxxednboosted')</script>` then whenever the page renders your comment to other users it will execute the arbitrary code.

DOM based XSS, unlike reflected and stored XSS, does not require interaction with a server in order to execute. The significance of this being that even sites designed to keep their backend protected from XSS are not necessarily safe themselves. The vulnerability for DOM based XSS is when the URL of a page represents some DOM elements. This way scripts can be injected through the URL into the DOM completely on the client side. An example of this would be a language setting such as `https://emurr.com/index.html?default=tagalog`. Such a URL can be edited to affect the DOM like so:

```
https://emurr.com/index.html?default=<script>alert('itshaxxintime');</s
cript>
```

This URL can then be sent to targeted individuals just like in reflected XSS as in phishing and social engineering tactics.

Mutation XSS (MXSS) is possibly the cleverest form of XSS out of them all and was the basis of the 2019 Google XSS vulnerability mentioned at the introduction of this topic. Through tools such as DOMPurify or Mozilla Bleach most XSS vulnerabilities are easily preventable, but not MXSS. MXSS relies on browsers parsing incorrect HTML and rearranging or mutating it into correct HTML. This allows the malicious scripts to bypass attempts to purify HTML by not being recognized as scripts. Once they have been accepted into the DOM they are rearranged into proper scripts and, of course, executed. For example, this is the script entered into the Google search bar that allowed our infamous bounty hunter to perform his MXSS attack: `<noscript><p title="</noscript><img src=x onerror=alert(1)> ">`. Although much more complicated than our earlier examples we can boil this down. The general overview is that the noscript tag, due to its improper formatting, mutates the DOM and the order in which the HTML is arranged. Furthermore, because the img onerror alert is wrapped in quotes it ends up passing the purification process. Ultimately, this results in arbitrary code that becomes greenlit by the purifier. However, when mutated by the DOM this code actually falls into the correct placement for the script to act upon rendering as in all other XSS attacks.

Cross-Site Scripting is especially meaningful for EMURR considering we have to consider not only HTML of our web builder but also the HTML of the pages that users generate which are then hosted on the Raspberry Pi.

The first consideration to make is how would we defend from XSS on our main web builder site? The most commonly suggested tactic for defending against XSS is to prevent input values like `<`, `&`, or `;`. However, this could get very tricky when we stop to consider our users are performing content generation for pages which become HTML. How, if we were to prevent special input values, are users able to specify tags in a way that is easy and intuitive? How do we give users the tools to make pages without also giving them the tools to perform XSS? Although not exactly clear, the EMURR team believes it is possible to develop a defense against this if we are careful about this security concern in our development process. Thankfully TinyMCE, the main tool we use to help support user generated content on the web builder, has a written article on security including XSS which gives us some direction on where to start [1]. At their suggestion we will explore integration with server-side HTML purification tools like DOM Purify. However, they admit that client-side XSS is still possible. We will have to be careful in how we structure the web builder in order to prevent vulnerabilities wherever we can.

These strategies help to defend our web builder from XSS but what considerations have been made about the tour sites? At their base design tour sites are static pages and therefore present no path to introduce XSS vulnerabilities. However, some of our stretch goals include making tour sites more interactive and therefore dynamic. When we arrive at that stage of development, we will have to review how best to defend against XSS spoiling some of our stretch goal features. Later, in our section on copyright we consider internalizing tour sites to be local to a physical EMURR unit. This helps ameliorate legal concerns and from the perspective of security would limit the scope of XSS. Under this model the only people with access to editing pages for EMURR would largely be grade school teachers. The likelihood of this demographic performing XSS of any form seems unlikely. However, it may not be good practice to dismiss attack vectors even if they are reasonably uncommon. With a localized model the only way in which an EMURR unit owner would be able to perform XSS on someone other than themselves is stored XSS. With this in mind, even under a closed model of EMURR it would be good practice to have server-side filtering such as DOM Purify.

2.4.1.2 NFC Insecurity

NFC tags have seen a long history but have only recently seen large scale commercial use and with that we wanted to share our understanding on what vulnerabilities this newer field may have introduced.

The main concern we had was whether or not someone could rewrite an NFC tag. Since the only function of an NFC tag was to point to a static IP the natural concern was whether or not someone could modify this information stored on the tag. As it turns out, most NFC tag designs someone can modify the data stored on it as long as they have an NFC enabled device. Since such devices have become so commonplace, namely smartphones, we were fairly worried about potential vulnerabilities introduced to the EMURR unit with NFC tags. Under this model a malicious actor could pass by the NFC tag and rewrite it to point to a site hosting malware or whatever else they desire. However, some NFC tags have been designed with a locking mechanism which we can make great use of to avoid this vulnerability. By flipping one bit on the tag we can guarantee that the rest of the information stored is unwritable by design. At first this seems like a loss in the ability to edit but with our design this does not present an issue. NFC tags simply point to a static, generic URL which will be populated when a tour pack is created, so there's no harm if the NFC tag is unavailable for editing. Although in security nothing is one hundred percent secure this seems to be an airtight design with little room for exploitation and we are excited to move forward with this design.

2.4.2 Data Privacy

If we can expect our users to come from anywhere in the world, we will also need to be mindful of data privacy laws globally. The most important examples of this are the California Consumer Privacy Act (CCPA) for users in California and the General Data Protection Regulation (GDPR) for users in the European Union. Both laws include various rights for the user, including the right to know what personal information is stored, the right to erasure, and the right to restrict processing of data. We will need to ensure our application protects the privacy rights of users whose data is stored on the application.

2.4.2.1 Data Privacy in the European Union

The rights afforded to European Union citizens by the General Data Protection Regulation include the right to be informed, the right of access, the right to rectification, the right to erasure, the right to restrict processing, the right to data portability, the right to object, and the right to object to automated decision making [2]. These rights are detailed below.

- The right to be informed: Users have a right to be informed about the collection and use of their personal data. This is usually fulfilled with a privacy notice.

- The right to access: Users have the right to request a copy of their personal data.
- The right to rectification: Users have the right to request inaccurate or outdated personal information be corrected.
- The right to erasure (also known as the right to be forgotten): Users have the right to request their personal information be deleted.
- The right to restrict processing: Users have the right to request the restriction or suppression of their personal data.
- The right to data portability: Users have the right to request their data to be transferred to another controller or to be provided to them in a format that a computer can use.
- The right to object: Users have the right to object to the processing of their personal data.
- The right to object to automated decision making: Users have the right to object to automated decision making based on their personal information.

2.4.2.2 Use of Cookies

Use of cookies is also governed by GDPR, the reasoning being that cookies are often used to track users and their personal data. Websites complying with GDPR must request explicit permission from the user before activating any cookies beyond what are strictly necessary for the site to function, such as to authenticate users who are logged in. Consent must be freely given, specific, informed, and explicit. Data subjects must be able to withdraw consent as easily as it was given.

Consent must be freely given by the user, and any inappropriate pressure or influence on the user renders the given consent invalid. Consent must not be tied to the quality of service received if the data processing is not essential to the performance of the service. In other words, a website may not deny or diminish services to the user for declining cookies that are not necessary for its function.

Consent must be informed, i.e., users should be made aware of what purposes the website is collecting cookies for. According to the European Data Protection Board (EDPB), users must be informed of the data controller's identity, the purpose for each of the processing operations for which consent is sought, the type of data that will be collected and used, the user's right to withdraw consent, and information about the use of data for automated decision making. Information must be provided in clear and concise language, which can be understood by the controller's intended audience. Consent may not be hidden in long privacy policies or general terms and conditions or written in complicated language.

Consent must be granular, i.e., users should be allowed to opt into certain types of cookies while opting out of others. For example, a user may opt out of cookies used for advertising but opt into cookies used for usage analytics.

Consent must be explicitly given. Cookies beyond what is necessary for the website's function should not be activated until the user explicitly gives consent to do so. Simply continuing to scroll or browse the site cannot be taken as consent to cookie usage. Cookies may not be enabled by default or pre-ticked in checkboxes prompted to the user. Acceptance of general terms and conditions cannot be seen as a clear affirmative action to consent [3].

In practice, many websites in compliance with GDPR prompt the user with a banner asking the user to consent to the use of cookies. The user may choose to accept all cookies or granularly decline cookies by category. Should this project use cookies in manners beyond what is strictly necessary – for example, to store preferences, or in case we integrate third-party services which use cookies – we will need to prompt users with a cookie banner informing them of our use of cookies and requesting consent.

2.4.2.3 Data Privacy in California

The rights afforded to California residents by the California Consumer Privacy Act include the right to disclosure, the right to receive personal information in a readily usable format, the right to opt in or out, the right to deletion, the right to not be subject to discrimination for the exercise of rights, and the private right of action for data breaches [4]. There are several overlaps with California data rights and EU data rights.

- The right to disclosure.
- The right to receive personal information in a readily usable format.

- The right to opt in or out: Businesses must include a link titled “Do Not Sell My Personal Information” in a conspicuous location on their website’s home page.
- The right to deletion.
- The right to not be discriminated for the exercise of rights.
- The right of action for data breaches: If a business fails to properly safeguard personal information and this results in unauthorized access or theft of such information, individuals can bring a private right of action with damages between $100-$750 per incident per resident or actual damages, whichever is higher.

2.4.2.4 Compliance

We aim to comply with the aforementioned data privacy regulations to all users on our site. Many sites apply data privacy rights of GDPR and CCPA to all users because it is simpler to implement, and because location data is not sufficient to determine whether these rights apply to a particular user. For example, CCPA applies to all California residents regardless of whether or not they are currently located in California [4].

To comply with users’ right to be informed, the website will contain a privacy notice page informing users of their data privacy rights, the information we collect, and how the information is used. Additionally, a cookie banner will be presented to users on their first visit, informing them of our use of cookies, a concise description of their purposes, and requesting consent for their usage. Additional information will be provided in a Cookie Policy page. Users will also be able to change their cookie preferences on this page, revoking their consent to our use of cookies at any time. We must also pass the user’s preferences to all third-party services integrated into the site to ensure they also do not make use of cookies not consented to by the user. The user’s preferences will be stored for no longer than 6 months, after which consent will be requested again for renewal.

To comply with user’s rights to access and data portability, the website will provide data export functionality on the privacy or user account page, where users can request their data in a computerized format.

To comply with user’s right to erasure, the user account page will provide a feature to delete their account, their created tour sites, or both.

To comply with other user rights, such as the right to rectification, the right to object, the right to restrict processing, the right to data portability, and others, the website will provide a contact form to bring their requests to the attention of an administrator.

2.4.3 Misinformation

As EMURR is an educational repository, we must consider the quality of informational content on the site, particularly if we allow editors to sign up as themselves and create their own tours from existing tour sites. Misinformation will be damaging to the integrity and reputability of the project. If untrusted editors are allowed to create tour sites, it is imperative we combat false information and defamatory content, particularly those of living beings.

A simple approach would be to implement a voting system, seen on platforms such as Reddit. The advantage of this would be minimal work from administrators in curating content, however, popular voting systems are not necessarily an indication of accuracy, so this approach is unsatisfactory and cannot be used in isolation.

Alternatively, we may implement a system for administrators to review and control content published in new tour sites. This can be achieved in one of two manners: a proactive solution in which new tour sites seeking to be published must be approved by an administrator (similar to requesting pull request reviews on GitHub), or a reactive solution where tour sites may be published but are subject to review and removal by an administrator if false information is found. These two systems are not mutually exclusive and can be made configurable by the user editing the page or the page that is being edited. The second approach can be improved by adding a reporting feature for users to report pages that contain false or misleading information to administrators, which will facilitate their job to curate and maintain the site. The downsides of these approaches are the requirement of administrator time and effort to read reports and new tour sites, and to independently verify the accuracy of new sites.

Many wikis use page or section locking or protection policies if sections are frequently edited with incorrect information, whether maliciously or because of common misconceptions, or to enforce copyright laws [5]. In particular, Wikipedia may block certain user accounts, IP addresses, or ranges of IP addresses, or protect certain pages or sections to various degrees, including holding edits by unregistered or newly registered users in review until approved by a reviewer or administrator. While the EMURR website is not a wiki, some or all of these features would need to be implemented to prevent forks of existing tour sites created by untrusted editors from routinely containing false information. All of the aforementioned approaches are not mutually exclusive, and a combination of approaches would serve to protect the website from misinformation.

Finally, publishing can be restricted to known and trusted users, such as those working at the UCF Center for Humanities and Digital Research. This will reduce the workload for administrators and ensure the quality of all publicly available articles on the EMURR site. Should our sponsors be interested in expanding this project, whether with a new team or as part of the stretch goals of this team, the design considerations above for fighting misinformation can be revisited and discussed in greater detail.

2.4.4 Copyright

2.4.4.1 Problem Overview

One requirement for EMURR is to display media such as images and video but with this requirement brings the burden of copyright infringement. When users generate their own content there is the potential for them to upload copyrighted media intentionally or unintentionally. While unintentional infringement is easy to fix with some moderation features intentional infringement isn't so much. Users intentionally committing copyright infringement will range from an apathetic user uploading a copyrighted image to a malicious actor attempting to use the EMURR system as a database for storing large amounts of media they do not own. Unfortunately, despite the severity of the vulnerability there is no silver bullet or step-by-step guide for every case of dealing with copyright. After explaining the problem to Sarah Norris, a scholarly communication librarian at UCF with a Master of Library and Information Science, we brainstormed some solutions.

2.4.4.2 Potential Solutions

One idea we have decided to not move forward with is to validate the identity of those editing the pages in the EMURR system. Essentially, upon sign up this feature would include tracking the user's Personally Identifiable Information (PII) such as a school ID, legal name, or address. With this information we would be able to properly assign responsibility to who is uploading media illegally, then report them swiftly. However, it does not take much imagination to envision how carrying PII turns into a slippery slope. Any worries we have about copyright which may be solved with PII will transfer into much greater worries about caring for sensitive user information. Any leak by software vulnerabilities, social engineering, poor operations security, or any other vector could lead to targeting of EMURR's users. Suffice to say, using PII to solve copyright is a dangerous game and the only way to win is not to play.

Another idea that was recommended against and put to rest was enforcing the Digital Millennium Copyright Act (DMCA). In order to do this, the team would first have to designate a DMCA agent which we would be responsible to report to. Next, we would have the burden for displaying DMCA information in an easily accessible manner on the EMURR site. After which the EMURR team would have to be willing and able to remove all infringing content as soon as we are made aware of its existence. Similarly, the team would need to be willing and able to terminate repeat offenders. The first drawback of this process is that maintaining legal reports to DMCA agents is tedious and bureaucratic on levels that our team is not prepared to handle. This is especially true given that EMURR was designed to eliminate bureaucratic processes blocking interactions with interpretive sites. The second drawback is that DMCA compliance takes manual review which we will not be able to support when this project is released as a finished product. There is potential to write automated tests for content but even then, such tests would be an imperfect solution. In conclusion, DMCA may be outside the reach for EMURR.

A possible way to shift the burden of copyright was to lean on online platforms such as YouTube to host videos. The intent was to have users upload YouTube videos and then embed them into their pages. However, the following is stated under Permissions and Restrictions:

> The following restrictions apply to your use of the Service. You are not allowed to:
>
> access, reproduce, download, distribute, transmit, broadcast, display, sell, license, alter, modify or otherwise use any part of the Service or any Content except: (a) as expressly authorized by the Service; or (b) with prior written permission from YouTube and, if applicable, the respective rights holders [6].

This plan would be an explicit violation of the YouTube Terms of Service and would land us in more trouble than before. Therefore, we have decided to pursue other options.

Internalized data is a strategy our sponsors are interested in pursuing which we are willing and able to deliver. Essentially, the plan behind internalizing data is to ensure that any user generated content is not publicly facing. The motivation for this strategy is so that any user generated content cannot reach others. Therefore, if any user generated content infringes on copyright it is at least not distributed to the public and remains local. The only downside to this strategy is that it creates dissonance with the intended concept for EMURR because it is meant to be a system to share knowledge and unburden teachers. The original idea allowed for anyone to become an editor and join the network without an EMURR unit. In contrast, with this internalization restriction editors would only be local to their unit. Furthermore, internalization introduces the downside of page redundancy. Separate users may have co-existing pages they individually created for the same interpretive sites. Where one user could have saved others time by writing and publishing pages this would not be possible under an internalized data model. A possible compromise is an approval system. If an editor wishes to share their changes they can request to publish and an administrator from or appointed by the CHDR lab will be able to review the page. If they feel the page is an acceptable representation of EMURR and does not infringe on copyright, then they can approve the page for publication. Under this concept it is unlikely for public pages to contain infringement and editors still get the opportunity to share their work.

It is clear in the matter of precautions that it is necessary to inform and warn the EMURR userbase of copyright concerns. Means of informing and warning users we have identified as valuable and effective are a signup agreement, post accompanied warnings, and informational pages. At account registration users will be prompted with a Terms of Service (ToS) agreement. In the Terms of Service users will read information concerning copyright concerns and how to appropriately address them in their own page creation process. The purpose of a Terms of Service agreement is to bring the users to an understanding upfront at signup. This way users know that although we host the site, provide physical units, and maintain a digital space it is up to the collective effort of individual users to adhere to policies in order to keep EMURR in a safe and stable state.

Furthermore, upon page creation users will be given a prompt to confirm their pages are either entirely original or that they have proper citations for what content is not their own. With a reminder at page creation, we make our agreement about citations obvious and explicit. Users will not be able to skip past Terms of Service or post carelessly forgetting about the agreement without citation. In respect to media, a possible feature the group has discussed is to require an included citation with the upload of any media on a page.

In addition to prompts, agreements, and upload requirements we intend to write publicly hosted and easily accessible pages connected to EMURR regarding our expectations on copyright. These pages will help to educate users on how to properly interact with the site and show our intent for user generated content with respect copyright concerns. Furthermore, we can use this as an opportunity to point users to public domain or fair use sources they can easily cite for their pages.

2.4.4.3 Fair Use

Wherever the EMURR system can stay within fair use we would be more than happy to do so. Fair use is content that is protected from copyright and is typically defined in four factors as stated by Cornell's open access resource on U.S. Law [7].

The first factor of fair use is purpose and character of the use, including whether such use is of commercial nature or is for nonprofit educational purposes. The fundamental purpose of this factor is used to protect education and commentary. We believe the primary purpose of the EMURR project is to promote education by allowing people to contribute to a digital mirror of physical interpretive sites. In this sense EMURR is positioned to argue for the first factor of fair use.

The next fair use factor is the nature of the copyrighted work. Essentially, this factor is referring to whether or not the work(s) being used are creative or factual in nature. For example, well known facts or non-fictional works such as a local news broadcast are likely to be seen as fair use. On the contrary, a work of fiction like a poem or a schematic are less likely to be seen as fair use. This factor of fair use is easily argued since EMURR is fundamentally for the purpose of sharing non-fictional data. However, if users can upload and share media apropos of nothing relative to the interpretive site this can hurt our chances here.

The third factor is the amount and substantiality of the portion of the copyrighted work to the whole. Small portions of copyrighted content are hard to justify as infringement because it does not lean on the sum of parts from the original work. For example, posting a six second clip from a movie will be much more likely to be judged as fair use compared to thirty minutes. This factor conveniently works well for EMURR because it is likely that the capacity of media held in a tour site is limited so that it can be loaded onto the Raspberry Pi. Therefore, enforced media limiting can hit two birds with one stone by adhering to both hardware and copyright constraints. With this in mind it would be difficult to structure tour sites in such a way to infringe on fair use factor three.

The fourth and final factor is the effect of the use upon the potential market for, or value of, the copyrighted work. This factor is an attempt to measure how much damage the new product causes, intentional or otherwise, to the existing products. Given user-generated content, it is entirely possible that EMURR can become a vector to share media in unintended ways. However, due to limited sizes and the small scope in which this will reach we believe that it will not be disruptive to any media industries such as film, music, digital art, and the like.

3 Goals and Objectives

Goals and objectives both refer to the desired outcome that our team wants to achieve to complete the project successfully and on time. The goal is a broad statement while the objective describes the methods used to achieve the goal. In other words, objectives are the steps the team needs to take to achieve each goal. Our team will strive for attaining the goals set forth herein and in order to accomplish this, we are proposing the following set of SMART (Specific, Measurable, Achievable, Relevant, and Time-bound) goals accompanied by its objectives:

- To complete and deliver to our sponsors a minimum viable product by the end of Fall 2022 Semester.
    - The web builder will have an easy-to-use WYSIWYG UI that allows non-programmer users to generate content for a tour and push the tour pack to a Raspberry Pi in a simple manner.
    - The web builder will have multiple user accounts so that different types of users can access the data and resources corresponding to their roles.
- To gain real work experience solving a problem to help bridge the gap between the digital infrastructure and interpretive information in public spaces for underfunded organizations.
- To build a cost-effective hardware solution that meets most if not all our sponsors' requirements that can be packed in a modular unit with minimal set up for the final users.
    - Design the EMURR unit to service not only one site. With minimal oversight and expertise, the EMURR unit could be easily changed to service multiple sites which will keep the cost low for educators and museums.
- To learn modern and emerging technologies in web development to keep up with trends in the computer science field.
    - Research technologies for web programming and select as a team the software stack for the development of the frontend, backend, and database of the web builder application.
    - Team members will familiarize themselves with the relevant technology according to the role assigned in the project.

- To enhance soft skills such as problem-solving, critical thinking, time management, teamwork, collaboration, and open communication all of which potential employers value in prospective employees.
- Hold biweekly meetings with our sponsors in Discord to perform a sprint review. Followed by sprint retrospective and sprint planning just among team members.
    - Sprint Review consists of discussing progress of the sprint, show results of the work, discussing upcoming milestones, and updating the product backlog.
    - Sprint Retrospective consists of discussion about what progress was made, what went right, what went wrong, what we could do to improve future collaboration.
    - Sprint Planning consists of planning the upcoming sprint, selecting tasks from the product backlog to enter the sprint backlog, and assigning them to developers
- During the weekly meetings team members will work collaboratively to achieve common goals, display progress, demonstrate positive attitude, and share information and resources.
    - Meet the project milestones set during the Sprint planning in a timely manner.

3.1. Stretch Goals

3.1.1 Introduction

Though the tour sites will primarily be built using the website builder, we proposed a few stretch goal features for the project. These are nice-to-haves but are not required to the overall intended functionality of our project. Features like commenting, heat mapping, and dynamic web applications can be used to enhance the experience behind the tour sites. Some of these features are intended to be built upon compile time while others are intended to be prebuilt and will be added to the tour site when requested. Building a React application purely based on a rich-text editor CMS will be difficult, but not impossible. Therefore, we need to take these stretch goals into consideration heavily, so we don’t over scope the EMURR.

3.1.2 Database

For the database, we have come across the idea of using a SQLite. It's a C-language library that is reliable, features like a SQL database, lightweight, and takes up relatively small amounts of memory. Setting up this database would be easy, and the syntax is similar to the MySQL database we currently use for the website builder. Additionally, if we store the comments in the main MySQL database, we can easily and reliably transmit the comment from the SQLite database to the MySQL database. We can then use these comments to lazy load in the website builder to show other tour guides the results behind attending the tours and provide unique experiences and user interactions for the next group who uses a particular tour site.

Furthermore, SQLite has plans for long term support until the year 2050. They do this by constructing their database architecture with cross-platform code and files. More so, the ability to share and move files from one database or filesystem to another should be unchallenging for our backend developers. SQLite includes testing, detailed documentations, and commented source code to aid developers in the process. Since this database architecture has already been established in software engineering, there are many other forms of support for libraries and plugins to interact with the database in the tour site API. Therefore, constructing a backend on the tour site should be a relatively intuitive process and could make for a relatively achievable stretch goal.

3.1.3 Comments

Comments will be built using a neat, lightweight database hosted on the Pi. During the consideration process of commenting on the tour site, we proposed storing comment data in a file. However, this does introduce race condition problems as the Pi will try to read and write to this file in the case of two separate tour attendees trying to access the commenting API at the same time. Luckily, lightweight databases are built around this very problem. Thus, it would be advisable to avoid reading and writing from a single file and use a local database on the Pi. Yet, this could propose other problems such as processing workload and other various performance issues. Through careful unit testing, we can test the number of users it would take to slow down the Pi with each API call and database call the API calls makes.

3.1.4 Heat Mapping

Heat mapping suggests interesting data for how the users interact with the tour. For example, if many students gravitate towards tomb stones or an art piece, we can infer, over a large data pool of multiple tours, that these points of interest gain the most traction. We can use these statistics to perhaps brighten the less popular attractions by stylizing their web pages in a more intriguing way or adding fun facts about them. In the cases of an art gallery, the host can use this data to, perhaps, take down a piece who gains no traction or enhance the pieces that do. See figure below for an example of how the heatmap may look at a tour location.

Figure 2: Example of a heatmap

Heat mapping should not be relative to the geography of the tour locations. Tagging a GPS would be too difficult for the scope of the project and would require too many outside frameworks to implement and manage. However, since we use NFCs to link to pages in the tour site, we can store this data fast and reliably in a one-to-one mapping. For instance, if one taps an NFC which redirects to a URL on the tour site hosted on the Pi, during page load, we can store a simple count in the database that represents a user interaction with said point of interest. This way, we don't have to overload the Pi with unnecessary location trafficking since we can infer the NFC location at the particular location. Then, if the organizations wish to review the statistical data, they can spot the location of the NFC in their exhibits or cemeteries and act accordingly.

3.1.5 Web Scraping

Most of the experience working with the EMURR unit will deal with the building of the website. An idea was proposed to implement a feature called pages. In short, pages are built to save the work of an individual who works on a web page on the tour site so that others can take their work and use it for their own tour sites. The purpose behind this is to alleviate redundancy upon information between two different tour sites, but that are attending the same tour. While this idea works well in practice, it only works on pre-existing tour site pages and doesn’t help the tour site pages that don’t already exist.

One stretch goal that was apparent to the success of our project was web scrapping. With web scraping, EMURR will scour the defined website programmatically and use the information to prevent our tour guides from doing the manual work of copy and pasting or even typing out the information themselves. The benefit of doing so saves time for our tour guides. Additionally, we can use the data from the website to appropriately cite the information presented from the distinguished website. Furthermore, we can ensure integrity by using direct quotes or minor alterations to the information provided. As discussed elsewhere, we need to minimize the amount of misinformation presented within our system.

Web scrapping will also alleviate the process of storing information in our database. Another idea that was proposed before pages was the ability to store information in the database to then use it as a part of the built website. The problem that persists is the liability of how accurate our information presents itself to the public, and therefore, we would be responsible for ensuring the information provided is correct. Thus, we seem it arbitrary to build a system like this and would rather have something along the line of web scrapping to achieve the same results, but to minimize the dangers of misinformation accountable to our project. In addition to web scrapping, incorporating pages almost achieves the same results, but doesn't necessarily apply to the validity of the information, just that we store it regarding the user's progress. In other words, we want to encourage our users to participate in the open web to provide the right information and give the correct citations for those sources through web scrapping.

3.1.6 Dynamic Tour Sites

Dynamic Tour Sites is arguably the most difficult stretch goal to obtain. We would have to create a custom compiler from our CMS to build a framework like React, Vue, or Angular to manage states and injections with JavaScript. Though this seems like a complex process, it's not impossible. Admittedly, this foresees the scope of the project as we would ultimately be creating another JavaScript framework to manage states and reactive components in the DOM. There may be a few other options for building on top of JavaScript frameworks such as React, Vue, and Angular, but even so would create potential problems in building a web application with something like NodeJS. Not to mention, we must compile, build, and run the respective commands on the Pi for this to ultimately work. Additionally, representing this system in our Next framework would also introduce potential problems. Compiling the websites tour guides write in the TinyMCE website builder to something like React could create potential problems and would take up much space in our filesystem. Therefore, we view the dynamic tour sites as a last resort and completely optional choice for our tour sites.

3.1.7 Conclusion

With all stretch-goal features taken into consideration, these are features that will contribute to the overall success and quality of life interfaces for the teachers / tour guides who will be interacting with our applications. As mentioned previously, we intend to implement a user story-based design. In other words, we wish to make the process of making, hosting, and interacting with the tour sites as intuitively as possible. All in all, the power to create the websites should be kept in the hands of the user under a strict and reliable rule set.

3.2. Requirements and Specifications

EMURR consists of creating the infrastructure for accessing digital assets on the ground in remote areas with weak/inexistent mobile network coverage. With that being said, our team and sponsor are aiming for the project to be an inexpensive and readily available alternative to overcome the economic barriers of smaller organizations and underfunded sites to create and propagate contextual digital media access.

The project will require a hardware infrastructure to host a tour pack and a wireless local area network to transmit the content on-site within a reasonable radius. A Raspberry Pi will host the content of a tour pack and a wireless router will broadcast within the outdoor area for the participants who will be able to connect their phones to the router's Wi-Fi. The router allows teachers and students on the field trip to access the website as they traverse the site/ground. Participants can also tap their phones on NFC tags scattered along the site that are programmed to URLs on the local Raspberry Pi to learn about the different spots on that particular tour area.

The requirements and specifications for the project as a whole as agreed to by both our team members and our project sponsors are listed below.

3.2.1 Hardware Requirements

- Raspberry Pi
    - The Raspberry Pi shall host a website for a field tour.
    - The Raspberry Pi shall run Raspberry Pi OS (previously called Raspbian) as the operating system.
- Wireless router
    - The wireless router shall provide Wi-Fi network and routing of localized resources.
    - The wireless router shall broadcast across a 600-foot diameter space as shown in Figure 2.
- Battery
    - The battery shall power up the 12V/5V setup while on the ground.
- Exhibit Markers

- To comply with the National Cemetery Administration, NFC tags shall be housed in 3D printed coin shaped enclosures and plastic garden stakes.
- Exhibit markers are not powered and can be placed at leisure around the site. For example, exhibit markers could be set up on top of the headstones in cemeteries.

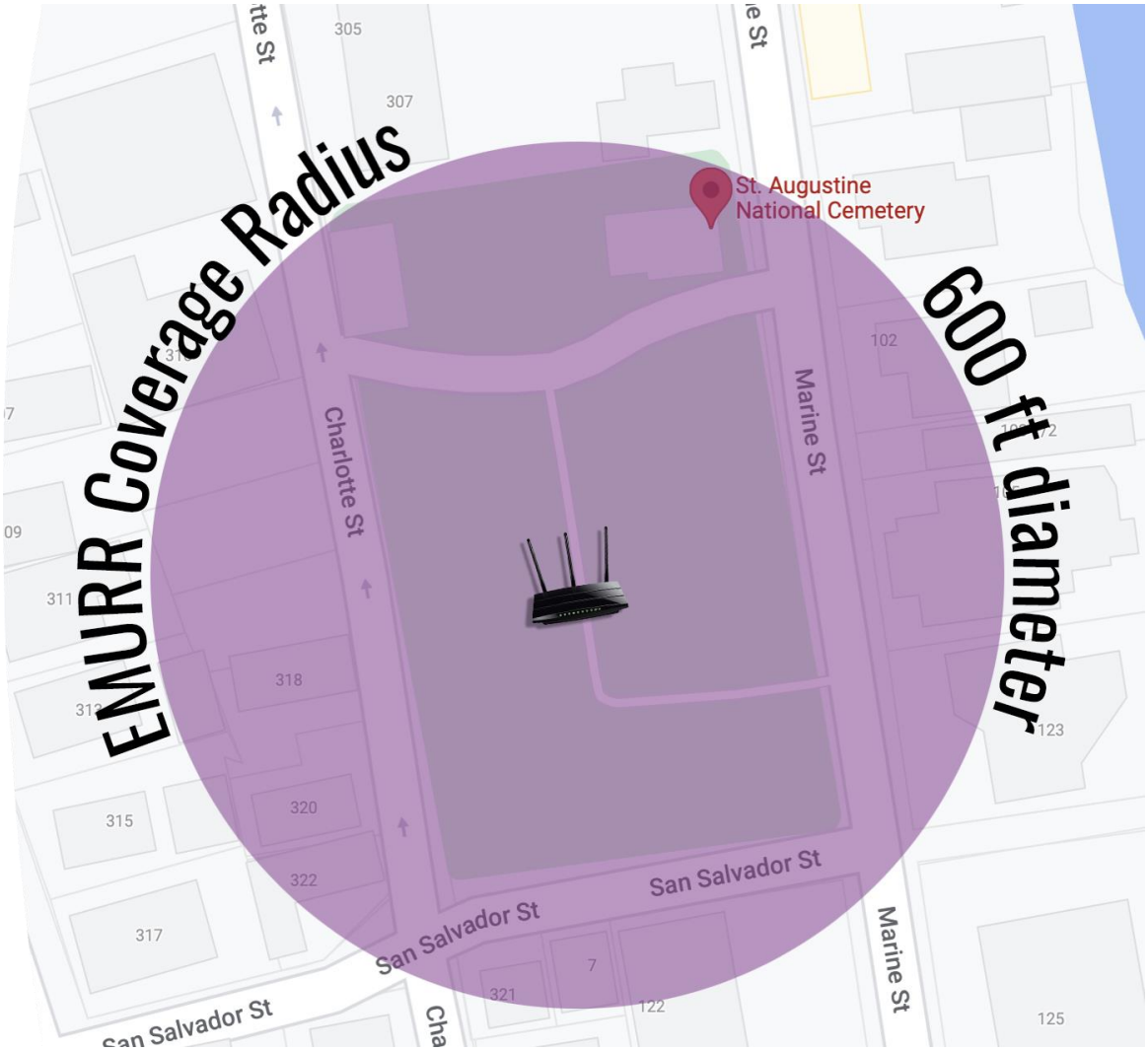


Figure 3: Example Exhibit Setup: St. Augustine National Cemetery

3.2.2 Software Requirements

- Database to store tour content and users' roles must be MySQL

- Administration website must have roles Super User, Administrator and Editor
    - A Super User shall be able to delete tour content, promote users between roles, and perform admin tasks on the router and Raspberry Pi.
    - An Administrator shall be able to build tour packs, push a tour pack to the Raspberry Pi and write the NFC tags.
    - An Editor shall be able to add content to a tour pack but not interact with the Raspberry Pi and the NFC tags.
- Tour site packs must be easily pushed to the Raspberry Pi through a web builder website UI
    - The web builder shall allow Administrators to create content for the tours.
    - The web builder shall provide a way to clear the Raspberry Pi from the previous tour.
    - The web builder shall provide an export functionality to push a new tour package to the Raspberry Pi.
- NFC tags need to be programmed (written) through the web builder UI
- The web builder needs to have a basic “What You See Is What You Get” (WYSIWG) editor to allow users to create content, including images and videos for a web page in the tour.
    - A local copy of the video shall be pushed to the Raspberry Pi and displayed with VLC or another Pi compatible video player.

In addition to the above-mentioned software requirements, the sponsors listed other functionalities for the web builder as stretch goals:

- PDFs or other ways for users to scrape existing web pages to put them into their tour along with citations to the original website. For example, Wikipedia passages, biographies, etc.
- Save the original URL and a citation for whatever was scraped in the database, which will not be pushed to the Raspberry Pi, but users will know where the content was taken from.

4 Ideas

As we discussed the implementation of this project, we saw fit the importance of user-story based design. This includes instructors, tour guides, tour attendees, and the website maintainers. Each of these individuals must contribute to the overall functionality of this project. The instructors / tour guides are responsible for hosting the website and relative API calls on the Pi. The tour attendees access the device through a browser on a mobile device or the like. The website maintainers must manage URLs on the NFC tags for end-to-end communication between the client and server. There are many ways to achieve the same end goals, yet some implementations of this project create more problems for the developers, instructors / tour attendees, clients, or website managers. Therefore, our goal is to design a framework for EMURR to balance the workload / setup for each group involved in the process.

4.1. Idea 1: Centralized Web Application

This includes maintaining websites upon changing edits, flushing out misinformation, and managing NFCs through a centralized organization (more on this later). As a result, creating websites corresponds to a WYSIWYG framework. While it does limit the extensibility and capabilities of our system, it generates simplicity and authority. Moreso, tutorials and guides used within our system should be easy to follow for our teachers / tour guides. Additionally, we can illuminate and embellish changes and features, allowing concurrent users to adapt easily to updates. We can extend this to our NFCs by managing all NFCs in our database. This will allow our users to easily attach a page to an NFC with a click of a button. Furthermore, managing misinformation is a breeze when controlling the publicity of a built website. The web application can incentivize correct information by limiting the (tour guide / teacher)’s ability to publish their websites to the public without being reviewed by one of our specialists first.

Pros:

- Minimal misinformation
- Websites are uniform and presentable
- Easy to learn and follow guides
- Cloud storage
- Prevent Griefing
- NFCs require no setup for tour guides / teachers

Cons:

- Limited functionality
- Requires foregoing maintenance
- Harder to contribute and share changes to an existing website
- Server management
- Consistent costs

4.2. Idea 2: Decentralized Desktop Application

The decentralized desktop application is responsible for managing the pipeline of transmitting cemetery websites to the Pi. This design was built with the instructor / tour guide in mind to reduce the workload of initial setup and interaction with the Pi. For example, one who sets up the server on the Pi would participate in downloading the desktop application, plugging in the Pi to a local desktop, finding the website online for the specific cemetery website they need, and clicking the upload button in the application. Everything else the desktop application should handle including errors between the Pi and the computer, faults in the website's code, and setting up the server locally on the Pi.

Pros:

- Instructor / tour guide user story requires minimal interaction
- Websites are extensible and decentralized
- Developing is easy and requires minimal effort
- Minimal to no cooperation with cemetery organizations
- Free

Cons:

- Storing data from clients and their interactions with NFCs requires additional features in the application, which incentivize not uploading data to the EMURR database
- Websites aren’t centralized to keep information accurate and useful
- Websites may be limited to static websites

4.3. Idea 3: Centralized Desktop Application

The centralized desktop application is responsible for managing the website repositories within our system. In this system, we incorporate our ideas from Idea 1 to streamline the workflow from building a website — to build for production — to upload the website to the Pi. Notably, handling URL requests may require a proxy which may also need to be implemented. This could be a complex process for our tour guides / teachers who don't work with server hosting on the regular. Therefore, we propose a centralized desktop application to handle server management. The user story of this application would require three interactions: connecting a Pi to a desktop or laptop, copy and pasting a unique identifier linked to the website they made; and clicking an upload button. Intuitively, the frontend UI is designed to make the process of server setup straightforward. On the backend, this application does many things such as: checks if the Pi is connected to a computer, checks the validity of a unique identifier, checks if the production build is correct, and manages the server by managing the file systems and routing proxies if necessary. Additionally, we can create error or successful messages to the tour guides / teachers to help guide them to fix smaller errors like wrong HTML errors, misinformation, and internal errors. Thus, this application is responsible for handling the frustrating hardware issues that can be difficult to debug for the average user.

Pros:

- Integrates in-house website application
- Helper application for complex processes
- Minimal server setup
- No interaction with Pi OS
- No file management

Cons:

- Changes on web application may need to also reflect on desktop application
- Balance cross-platform availability
- Creates another software application learning curve

5 Research & Investigations

5.1. Hardware Specifications

As per the requirements, the Exhibit Marker Uniform Resource Router (EMURR) project should have a hardware infrastructure to host a tour package website and broadcast the information contained in the tour package to remote outdoor areas with limited access to the internet. The main components of the hardware infrastructure are the following:

- Raspberry Pi for web hosting
- Router to create a wireless local area network
- NFC Tags to write the tour package URL and launch the website when tapped.
- Battery to power up the setup

In addition to the hardware components mentioned above, the setup will require a microSD card for the Raspberry Pi with a minimum of 128 GB of storage, and power and connection cords.

5.1.1 Complex Single-Board Computer: Raspberry Pi

A Single-Board Computer (SBC) is a full-fledged computer built on a small single board. SBCs typically include a processor, memory, and input/output ports. They are different from standard computers in several aspects:

SBCs are smaller in size given that all the main components are on a single board making them more compact and lightweight.

- SBCs consume less power and are more affordable as they are not equipped with extra components for basic operations.
- SBCs are easier to set up as everything is built onto a single board

SBCs can run a fully functional operating system such as Linux, Windows, or Android and are very versatile to support a variety of tasks; including, hosting small servers and databases and light web browsing. There are several manufacturers and examples of SBCs being Raspberry Pi (RPi) one of the most popular in the market.

A Raspberry Pi is a series of low-cost single-board computers the size of a credit card that was developed in the United Kingdom by the Raspberry Pi Foundation, an educational charity to promote the education of children and adults in the field of computer science and related subjects through the power of computing and digital technologies [8].

Raspberry Pi features USB ports for connecting peripherals i.e., standard mouse and keyboard, HDMI ports to plug into a computer monitor or TV, and multiple General-Purpose Input/Output (GPIO) pins to serve as a physical interface between the Pi and the outside world, which allow connecting to external devices such as LEDs, cameras, temperature sensors, and many others.

There are three series of Raspberry Pis and each series has several generations:

- **Raspberry Pi**

The first generation of Raspberry Pi was released in 2012 with the Pi 1 Model B featuring a single-core 700 MHz CPU and 256 MB RAM. Since then, there have been several iterations of this series from Pi 1 to Pi 4 with a 1.5 GHz 64-bit quad-core CPU, and 1-8 GB RAM. Generally, most series have Model A and Model B. Model A is a cheaper variant due to it tends to have reduced RAM and fewer ports. The latest release is the Raspberry Pi 400 featuring the same CPU as Raspberry Pi 4 but built into a portable keyboard.

- **Raspberry Pi Zero**

It is a spinoff of the Pi 1 with a smaller size, cheaper, and reduced GPIO pins released in 2015. The latest iteration of the Pi Zero series is the Raspberry Pi Zero 2 W launched in 2021 which is based on the Pi 3. Contrary to the prior models (Pi Zero, Pi Zero W, and Pi Zero WH), it is 64-bit capable. The Pi Zero 2 W is the model selected for the EMURR project; therefore, its specifications will be presented in the next section.

- **Raspberry Pi Pico**

It is the first board based on a single microcontroller chip, the RP2040 features a dual-core CPU with 264 KB RAM and support for up to 16 MB of off-chip flash memory. Released in 2021, the Pico is tiny, fast, and versatile. Its cost is around $4.00.

5.1.1.1 Pi Zero 2 W, Pi 3 Model B+, and Pi 4 Model B

Taking into consideration that one of the main objectives of this project is to offer a cost-effective solution for underfunded community groups and organizations, our sponsors favored the Raspberry Pi Zero 2 W for the hardware infrastructure which cost is listed at $15.00 in the official Raspberry Pi website. Under normal circumstances, an EMURR unit with a Pi Zero 2 W, a set of 15 exhibit markers, and accessories will cost in the range of $200-$300 dollars.

At the time this document is written, there is a global chip shortage crisis in which the demand for integrated circuits exceeds the supply chain, and Raspberry Pis are not immune to it. With the expectation that the supply chain challenges will continue through pretty much 2022 and a significantly increased demand for Raspberry Pis, acquiring any single-board computer in the present climate is an arduous if not impossible task. The prices have increased considerably, for example, a Raspberry Pi Zero 2 W is being sold on sites like Amazon and eBay for up to $120.00, eight times more than its manufacturer's suggested retail price. The increase in price is transitory and as the global supply chain of semiconductors moderates, we expect the Raspberry Pis to get pricing back to where it was.

We estimate an EMURR unit will suffice for a group of 15-20 people, but if the tour group size increases or the digital assets include binary data files such as images and videos, a more robust hardware setup might be required. Hence, despite the requirement of using a Pi Zero 2 W, our team will evaluate other possibilities of upgrading the computing power by choosing a Pi 3 Model B+ or a Pi 4 for the EMURR unit.

Table 1 below shows the relevant hardware specifications of the Pi Zero 2 W secured for the project and the other two models being considered, Pi 3 Model B+ and Pi 4 Model B.

Specs	Pi Zero 2 W	Pi 3 B+	Pi 4 B
CPU	1GHz 64-bit quad-core processor	1.4GHz 64-bit quad-core processor	1.5GHz 64-bit quad-core processor
RAM	512MB	1GB	1GB, 2GB, 4GB, or 8GB
Wi-Fi	Yes (2.4GHz wireless LAN)	Yes (2.4GHz and 5GHz wireless LAN)	Yes (2.4GHz and 5GHz wireless LAN)
GPIOs	HAT-compatible40-pin I/O header footprint	Raspberry Pi standard 40-pin GPIO header	Raspberry Pi standard 40-pin GPIO header
Ethernet Port	No	Gigabit Ethernet	Gigabit Ethernet
USB Ports	1×micro-USB 2.0 interface with OTG	4×USB 2.0 ports	2×USB 3.0 ports. 2×USB 2.0 ports.
Storage	microSD Card up to 256 GB	microSD Card up to 512GB	microSD Card up to 2TB

Table 1: Raspberry Pis specifications and comparison

Figure 4: From left to right, Pi Zero 2 W and Pi 3 B+

The main limitation of using a Raspberry Pi Zero 2 W is that it only supports 2.4 GHz Wi-Fi with a real-world bandwidth of around 36 Mbps [9]. The average bandwidth required for streaming HD video (720p) is 2.5 Mbps while the bandwidth required for streaming Full HD video (1080p) is 4 Mbps. Assuming the worst-case scenario of 20 users connected and streaming video at the same time, the minimum bandwidth needed would be 50-80 Mbps depending on the video quality. According to this analysis, the Raspberry Pi Zero 2 W would only be suitable for a smaller group. Raspberry Pi 3 B+ and 4 B are the two hardware options to be considered.

5.1.1.2 Operating Systems for Raspberry Pi

There are many operating systems available for Raspberry Pis. Raspberry Pi OS and Ubuntu are the two most popular. They are both based on Debian and share many similarities with the base system. Table 2 shows the options to consider for this project.

Operative System (OS)	Version	Compatible with
Raspberry Pi OS (32-bit)	RPi OS with desktop RPi OS with desktop and recommended software RPi OS Lite	All Raspberry Pi models
Raspberry Pi OS (64-bit)	RPi OS with desktop RPi OS Lite	3 B+, 4, Zero 2 W
Raspberry Pi OS (Legacy)	RPi OS (Legacy) with desktop Raspberry Pi OS Lite (Legacy)	All Raspberry Pi models
Ubuntu	Ubuntu MATE	3 B+, 4
	Ubuntu Server Ubuntu Core	3 B+, 4, Zero 2 W

Table 2: Operative systems for Raspberry Pi.

Ultimately, after researching Raspberry Pi models and operating systems and weighing the requirements of EMURR, we recommend a Pi model 3 B+ running Raspberry Pi OS (32-bit or 64-bit). Pi 3 B+ is the most cost-effective option for hardware and Raspberry Pi OS is the best overall OS for Pis based on compatibility, updates, installation, applications, and performance.

5.1.2 Router

One of the key aspects of the project relies on the use of a router to distribute a teacher created tour website around the area being visited. Most popularly known for their use in local home networks, routers have the primary job of creating and transmitting a network that multiple devices can connect too. They also serve the function of being a packet forwarding device for their given specified network. More specifically, they forward received data packets from external networks to devices located on the local network being broadcasted.

As previously stated, routers primarily communicate through the exchange of data-packets. These packets often consist of transmissions of information or requests for information. To be more concise, these packets can carry information such as protocol requests between a network and a device, text or email messages, voice or video information, and even streamed entertainment services such as video or music. These packets don't just contain the data being transmitted, or payload, however.

In order for the router to know where the packets are being sent, these packets also contain 'headers' that store the destinations IP address. These headers are also known to contain the packets origin IP address and the order separate packets were sent in.

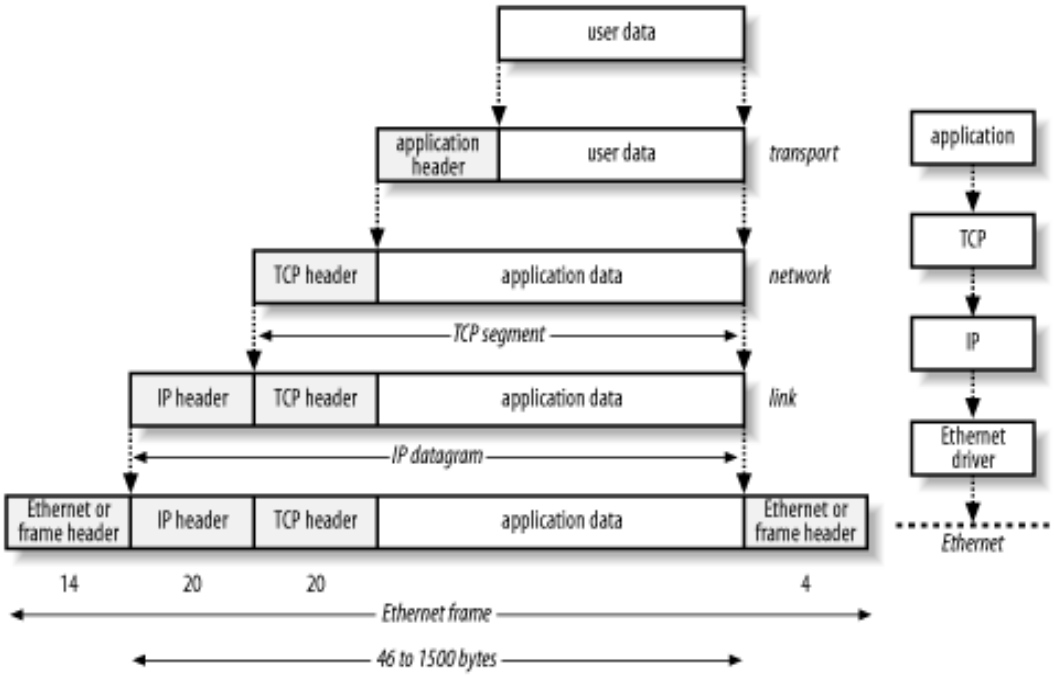


Figure 5: Diagram of IP Packet [10].

When a router receives any number of packets, it performs two specific functions. The first function consists of reading the packets header and locating the destinations IP address. The second function then relies on determining the quickest way to reach that specific IP address and then rerouting those packets to the correct location. To perform this task, routers use what is known as a routing table. A routing table is a location table that is built into the router and contains all the devices connected to its network, their IP address, or the quickest way to reach each device. So, once a packet is received by the router and its destination IP address is read, the routing table is checked and used to guide the packet to its next location.

5.1.2.1 Types of Routers

Nowadays, routers come in various forms, all customized for specific functions. After determining a router would be required for this project, the various types of routers available were researched to decide which would satisfy all requirements the best. Below are some considerations taken by our sponsors when deciding what router would be the most suitable to complete our task.

The first consideration was whether or not the router should be wired or wireless. The main difference found between both options is that wireless routers transmit received packets to devices on the network through radio signals. Wired networks alternatively send packets through wires directly connected to devices on the network. It was known that wireless functionality would be needed to transmit the tour website from the Raspberry Pi to the NFC tags. Due to this, our sponsors opted to go with a wireless option that also had a USB port in order to allow connection to the Raspberry Pi.

The next major consideration was whether or not a router-modem single device would be needed. A modem is a device that has the ability to connect to an Internet Service Provider, or ISP, with the sole purpose of being granted access to the internet. To transmit this access, it receives to other devices however, and preferably wirelessly, it has to be connected to a router. Due to the convenience of having a single device as opposed to two devices, a majority of routers that can be found are 2-in-1 router-modem combinations. These options due come with some major drawbacks, however. Usually, these alternatives have lower performance than their separated counterparts and to achieve comparable results, more expensive versions have to be bought. It must be noted however, that these 2-in-1 options are usually cheaper than buying two standalone devices, but that comes of the price of sacrificing scalability as upgrading components over time is

usually cheaper than buying another 2-in-1 device. Since it was known that our product was going to be primarily used in offline areas, the purchase of a 2-in-1 device was deemed unnecessary, however this came at the cost of a potential security fault that is discussed in the next section.

5.1.2.2 Security Concerns

Albeit few, there are security concerns that exist when using a router. The first, and least concerning to our project, is the capability of DDOS attacks occurring. DDOS attacks stand for Distributed Denial-of-Service attacks. These attacks occur when an individual or company floods a router with a large number of requests. Due to the quantity of requests, a router being used can become overwhelmed and crash. Taking the scale of our project into consideration however, we do not see this as a major concern, as we do not think anyone would commit this attack on the router. Additionally, the need for a router with the ability to handle a large number of requests, for example in the case of large tour groups, was emphasized during the purchasing process.

The second and most relevant concern when it comes to use of a standalone router, however, is security openings as a result of out-of-date firmware. Most routers perform firmware updates automatically as a result of having direct access to the internet. Since our router is primarily going to be used in offline situations, and most likely never be connected to an internet source, there is a real concern of its firmware constantly being out-of-date. This is a problem as outdated firmware does not contain potential security fixes that may have been posted. If security fixes are never downloaded, then openings are available to disrupt connection and access to the tours. Due to the potential of this problem, our group is creating a solution that instructs users to periodically allow the router to have access to an internet source in order to have needed updates be performed.

5.1.2.3 Our Choice of Router

Our sponsors have currently elected to go with the TP-Link Archer C1200.

Its hardware specifications consist of:

- A Single-Core 900 MHz CPU.
- 16 MB Flash Storage.
- 128 MB of RAM.

- 4 Ethernet Ports.
- 1 USB Port

Its software specifications consist of:

- Range of 150 meters.
- IEEE 802.11ac/n/a 5GHz Wireless Standard.
- IEEE 802.11n/b/g 2.4GHz Wireless Standard.
- Speeds up to 867Mbps for the 5GHz channel.
- Speeds up to 300Mpbs for the 2.4GHz channel.

This router also offered significant benefits with a miniscule number of disadvantages. They are listed below:

Pros:

- Speeds are more than sufficient for hosting purposes.
- Size allows for easy transportation.
- Simple administration.
- Lightweight.
- Inexpensive.

Cons:

- Need of external battery adds extra weight and can prove to be uncomfortable.
- Range might be too small for our intended purpose.

For the purposes of being able to broadcast the local site hosted on the Raspberry Pi, it appears the router is more than sufficient for satisfying that need. Furthermore, the scenario in which a large number of users are required to connect to the router also appears to be handled, as the router should be able to handle a significant number of users being connected to it simultaneously. Altering and accessing administrative details is also simple as standard methods and user-friendly interfaces are provided for easy customization.

The issues presented with this specific router, however, mainly consist of the need for an external power source adding an undetermined amount of weight to the amount being carried and the routers range being too small for the projects purpose. Beginning with the need for an external power source adding weight to the router as a whole, the inclusion of a 5V battery can add a significant amount of weight required to be carried by the teacher themselves and this can grow to be uncomfortable over large spans of time. The most significant issue, however, consists of the fact that the devices range has the potential to be insufficient for large scale fieldtrips. With a range of 150 meters, our router should be more than sufficient for uses in small museums, local cemeteries and in situations where students are not permitted to travel far from their supervisor.

However, given a situation where a location exceeds the routers range and students are not required to be in direct vicinity to their teachers, it is not only possible that students will not be able to connect to the router, but also that NFC tags too far from its range will also not be able to connect. Even given a situation where a locations area is within the devices range, there is still the possibility of a teacher being on one extreme of a location affecting connection on the other extreme.

To counteract the determined issues, our group has begun looking into high-capacity lightweight batteries as well as purchasing Wi-Fi extenders that can be placed in planned locations to ensure that connection is available throughout the entirety of a location.

Figure 6: Front and back image of the TP-Link Archer C1200 showcasing ports and power connections [11].

5.1.2.4 Testing the Router

In order to ensure the router is sufficient for the completion of the project, we have developed various testing strategies to ensure it meets needed requirements. Most tests require a test website to be loaded onto the Raspberry Pi and the use of an NFC tag that has access to a page on the site. Due to this, an initial step when it comes to testing our router is to create a temporary website and push it to the Raspberry Pi. Afterwards a page on this site must be assigned one of the URL's found on the tags.

The first test we want to conduct is making sure an ample number of devices can communicate with the router at once, so we plan on connecting all members phones and laptops to the router and seeing if performance suffers after a given amount.

The second test consists of making sure the range on the router is sufficient for our intended use. To do this, we plan on having devices connect to the router and progressively get farther and farther away from the router itself. We would then measure the distance at which the connection to the router becomes poor and when the devices begin to lose signal from the router in general.

Our final two tests consist of determining how long our picked battery grants the router power and how long a person can carry the router and battery before it begins to become uncomfortable. For the first test, we plan on fully charging the battery, leaving the router on and simply calculating how long the router has power. For the second test we will have multiple group members transport the router, battery, average sized laptop and minimal sized personal items throughout an average day at campus. This is due to our group estimating that is what an average tour guide will be carrying around a given site. Based on results, the possibility of acquiring a new router or looking into a more lightweight battery may become alternatives.

5.1.2.5 Deciding to Upgrade Routers?

At the moment there is no decision to upgrade to a different router. Primarily based off the range test, if we discover that in order to reasonably host a tour, a router with more range is needed then we may decide to purchase an alternate router. However, due to cost, it may be more feasible to buy Wi-Fi extenders.

Figure 7: From left to right: TP-Link N3000 Wi-Fi Extender that was considered; TP-Link AC1750 Wi-Fi Extender that was considered.

5.1.3 3D Printing

A major technological piece of hardware that was utilized throughout the creation of our project was 3D Printing. Most, if not all pieces of hardware utilized the technology in one way or another and a lot of focus was placed in designing add-ons that would ease the distribution of either NFC Tags or aid in the movement of the router.

5.1.3.1 What is 3D Printing

3D Printing is the process in which a material, usually silicone, is used to create a physical version of a 3-dimensional design created by an individual. The general process commences with any given type of plastic, being heated to a degree where it begins to liquify. This machine then maintains the material at this temperature till the point where it is being laid down on a printing surface through an automated arm and printing tip. Once printing begins, the material is usually laid in multiple layers and at an exact temperature where once dispelled, it quickly hardens. The machine also prints each layer at such a speed where each layer has an ample amount of time to solidify before the next layer is applied. Due to this requirement however, the process can generally, last multiple hours depending on either the complexity of a build, quality of the machine, or even the material used.

In order to be able to physically print something, a design of the object has to first be imported or created in specialized and separate software. These pieces of software usually allow the capability of directly creating a 3-dimensional model of what is to be printed, or importing a design made in other software directly. Once imported, the software then usually implements support beams based on the distribution of material throughout a design. These support beams can generally be removed easily but are included in order to allow for the printing of complicated designs.

5.1.3.2 Why Was 3D Printing Used

Our sponsors elected to go with 3D printing as the means of production for our needed custom components due to an array of reasons. The primary reason consisted of 3D printings' cost effectiveness. Due to printing facilities being directly available to the group at the university, and the existence of third-party 3D printing companies in the area, 3D printing proved to be a very affordable option despite what was initially thought. It is true that the machines generally cost thousands of dollars to attain, but due to the low cost of operation, most companies and our own campus provide the ability to mass produce components at a low price and with quick turnaround times. Because of this, 3D printing was heavily favored when considering various options.

Following cost, 3D printing as utilized due to its customizability in designing prints. The pieces that were required to be made for both the router used and the NFC tags purchased, were parts that either did not exist or were too specific to be purchased elsewhere. An example includes the arms attached to the router, which allow the router to be hung on the back of bookbags. Arms such as these are not available for purchase in third party stores and any possible alternatives would either need to be modified heavily to be applied to our routers specifications or would lack security. 3D printing essentially fixed all these concerns as the access to 3D printing allowed the development of materials made exclusively for the equipment purchased. Through simply measuring our own tools, add-ons with exact dimensions were able to be designed and printed rapidly.

Aside from the access to custom add-ons, the ready accessibility most schools and individuals have to 3D printing is a factor that played precedence in our decision. As 3D printing has become more commercially available, most people, including K-12 educators can find companies that will 3D print parts for them in a timely and affordable manner. More so, having consumers of our product have their own parts produced is an option considered solely as a worst-case option. If a sizeable number of parts need to be made, there is access to mass production in order to fulfill any potential orders.

The final motivation for choosing to utilize 3D printing in our project is the fact that our sponsors are very familiar with the practice, and design 3D prints regularly. More specifically, our sponsor Evan Wallace designed the carrying components for the router as well as the coins used to carry the NFC tags. Due to the experience available, 3D printing is heavily viable option as most ideas can be designed and produced within our own team. Since the involvement of our group in the project, new designs have been suggested and made easily, so our accessibility to 3D printing has proven to be very reliable.

5.1.3.3 Machine Used

For our project, the designs made were printed using a Makerbot Replicator 5th Generation. This machine is a mid-range printer made available through third-party printing companies, as well as through on campus facilities. Its physical specifications are listed below:

- Type of Material Used: Polylactic Acid Filament (PLA).
- Workspace Dimensions: 9.92” x 7.83” x 5.9”.
- Average Operating Temperature: 60-90°F
- Physical Connections: USB/Ethernet.
- Power Necessary: 100-240V.
- Filament Diameter: 1.75mm.
- Nozzle Diameter: 0.4mm.
- Printing Speed: 150 mms/s.
- Number of Extruders: 1.

Its software requirements and features are as follows:

- Required Operating Systems: Mac OS X (10.9+), Windows (7, 10)
- Primary Software Used: MakeBot Print Software.
- Wireless Connection: Wi-Fi Capable.

- File Types Accessible: STL, OBJ.
- Print File Type: MAKERBOT.

Due to the accessibility to said machine and the low level of complexity of all our prints produced, this machined proved to be more than ample for our purposes. All objects being printed were of a generally small size and could be compressed, so the housing size where components were actually printed was sufficient.

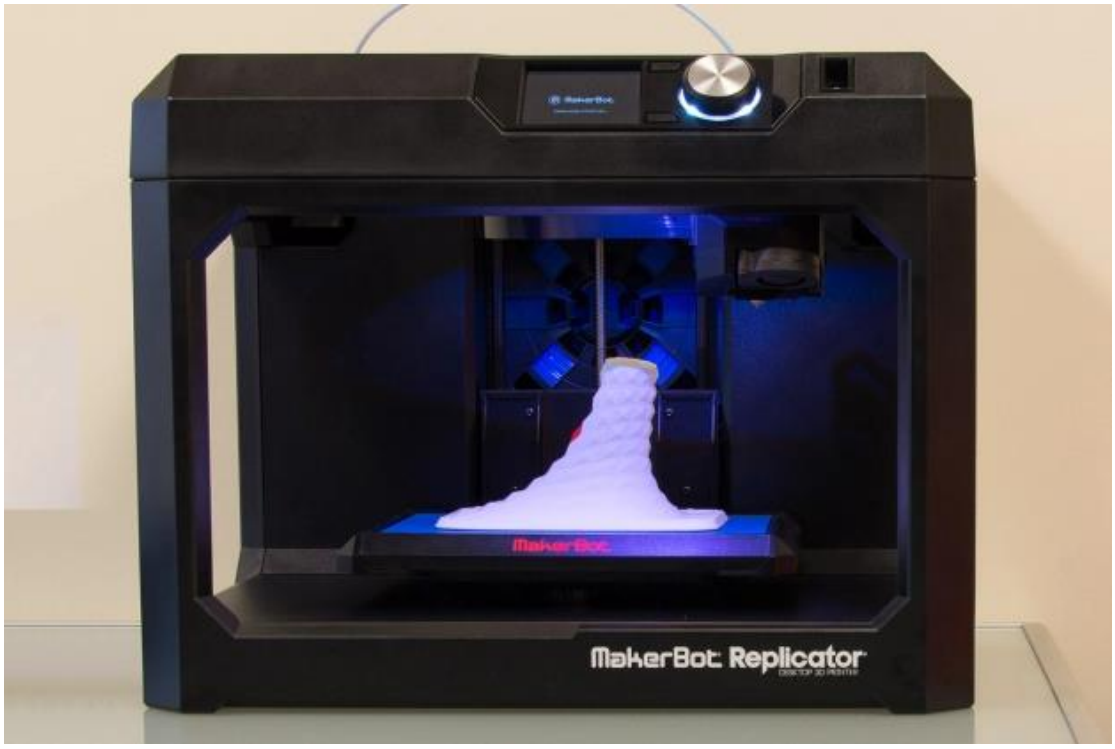


Figure 8: Photo of MakerBot Replicator 5th Generation Used.

5.1.3.4 Material Used

All prints produced were made out of 1.75mm polylactic acid (PLA) filament. This filament was used in particular primarily because it is the main material used by our sponsor and by the third-party vendors where parts were ordered.

With that said however, this material was home to a wide array of benefits and advantages that heavily aided and satisfied the requirements of the project. To begin, this material is known to have a low operating temperature which eases various parts of the printing process. For example, its low melting point allows for lower operating costs when running a machine and preserves machine components for a longer period of time. This is heavily seen in less filament build-ups occurring while printing due to the material being easier to melt. Lower temperatures do not just preserve the printer however, materials that generally require less heat tend to retain their shape and avoid warping during the printing process due to the extruder operating at a lower temperature throughout the print.

The following benefit to using PLA in contrast to other materials is its durability as a material. For our purposes we needed 3D prints that could handle frequent use and movement, as well as outdoor exposure. Using PLA provided this benefit as the material retains its shape well and can sustain conditions where the prints are normally used. If the printed components are dropped or stepped on, the amount of damage seen will be minimal which is vital in the context of our project as our devices will be used heavily outdoors. The material can also be altered significantly as its surface allows for easy application of adhesives, drilling, or even sanding Therefore the prints can be modified given a specific circumstance instead of having to be reprinted.

The final perceived benefit of using PLA is that it is biodegradable. Albeit not a significant point, our group was heavily in favor of using this material when we found out it had the ability to be recycled and biodegrade. Given our original concept of using plant stakes to scatter around NFC coins, we found it beneficial that our components would not cause extensive environmental harm if lost.

5.1.3.5 NFC Tag Components Produced

What was needed in regard to our NFC tags was an encasing to protect and hold the tags, while also having the ability to be easily attached or adhered to surfaces. For this design, a coin that a tag could be inserted to was made. The coin is bigger than that of a tag but was kept small enough to avoid issues when attempting to find locations for the coin to sit. Furthermore, the back of the coin allows for adhesives to be attached to it in the case that it needs to be placed on a wall or other surface.

Figure 9: Image of the 3D NFC Tag Component that was printed. The top portion was printed, and the tip is the coin design that contains an NFC Tag.

5.1.3.6 Router Components Produced

Figure 10: 3D Printed Arms attached to the side of the router.

The components that were made for our router, solely consisted of two arm attachments meant to be able to hang the router to a given tour leaders book bag. Both of these arms attach to the side of the router and have hooks that allow for the attachment of the router to the outer straps of a bag. Furthermore, each arm has a socket on its side for a string to be run through it, in the case that the router needs to be further secured. The components are also lightweight in nature and do not contribute any significant weight to the router in question.

5.1.3.7 Potential Problems with 3D Printing

Despite integrating 3D printing into our project, there are some potential areas of concern that our group addressed in regard to its use. The first of these consists of most prints not being waterproof. In the case of our router add-ons, this is not really a concern as the router should be in a dry location throughout its use, but for the NFC tag encasings, this vulnerability could prove to be an issue. Primarily due to the fact that these coins have the potential to be placed in any location, if poor weather conditions occurred while the tags are distributed, there is a chance that some could be damaged.

An additional problem that was discovered is that replacing damaged or lost components can prove to be a timely an arduous task for those who will acquire one of our sets. Because the pieces for the equipment are custom made if a user damages or loses a component, there are little to no alternatives for getting replacement pieces aside from contacting us directly. This could be a very time-consuming process as new components would have to be ordered and assembled on top of shipping time.

The final issue found revolves around not having direct control over the quality of the prints produced. In the situation where various components need to be mass produced there is a possibility that the quality of the prints could fall with an increase in production. This would primarily be due to hardware degradation, but depending on the scope of the project, this could be a significant deterrent.

5.1.3.8 Alternatives to 3D Printing

Despite having components already produced through 3D Printing, our group has considered alternatives for encasing and augmenting our existing hardware. These considerations primarily revolved around the protection of the NFC tags, as finding alternatives to the router was not deemed necessary. The initial idea of protecting the tags was to encase them in acrylic as shaping and cutting acrylic is an accessible and affordable option. A lack of experience of work with acrylics on our team however, made this option more trivial, but still an alternative. The next option deduced was simply placing the tags in small and sealed plastic bags, in order to ease delivery and production. This option proved poor however, as the tags could be much more easily damaged accidentally, there would be no viable way to adhere them to most surfaces and, they could be misplaced if not kept careful track of.

5.1.4 NFC Tags

5.1.4.1 How do NFC tags work?

A Near Field Communication (NFC) tag is a small, unpowered device that interacts with a powered NFC-enabled device such as a smartphone to exchange information [12].

An NFC device communicates with the tag by generating a radio frequency (RF) field. The RF field supplies power to the tag through magnetic induction, the same way wireless charging works, and carries communication between the tag and the device. Because the tag receives power from the device, the technology only works over short distances, roughly 4 inches (100 millimeters) or nearer.

NFC is commonly used for contactless payment methods, most notably through digital wallets such as Apple Pay or Google Pay. The technology has rapidly increased in popularity in recent years, in part thanks to the COVID-19 pandemic and the increased need for contactless payment.

The NFC protocol allows for two-way communication, which enables transactions that are dependent on data from two devices, such as card transactions. Some tags also include advanced features such as read-only locking, which will be helpful for protecting our tags from modification from malicious users.

5.1.4.2 Specifications

The NFC tag model selected by our sponsors is an NTAG 215 which is the midpoint tag between a 213/216. It has 504 bytes of storage data and is based on the NTAG21X standard.

In total, the NTAG215 EEPROM model contains 540 bytes, organized in 135 pages of 4 bytes per page. 26 bytes are reserved for manufacturer and configuration data, 28 bits are used for the read-only locking mechanism, 4 bytes are available as the capability container, and 504 bytes are available as user programmable read/write memory [13].

The model also contains a number of security features. Most importantly, the tag can lock sections of its programmable memory as read-only, which will prevent any visitor with an NFC-enabled device from writing to the tags.

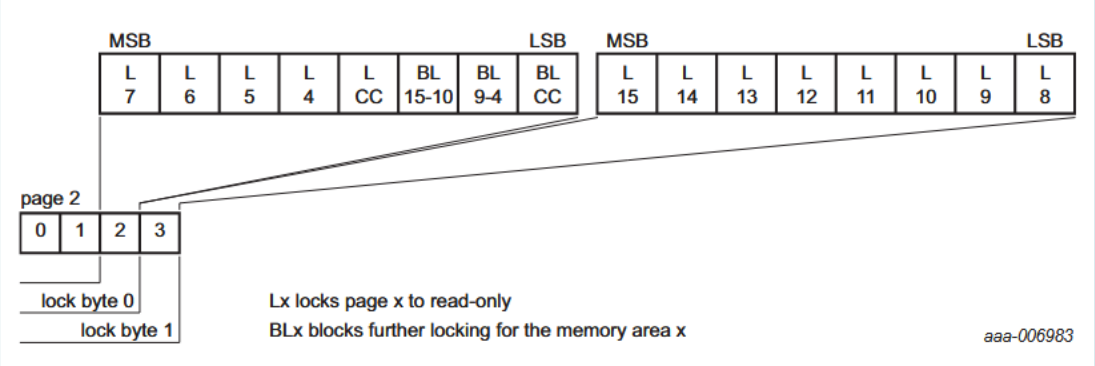


Figure 11: Diagram for static locking

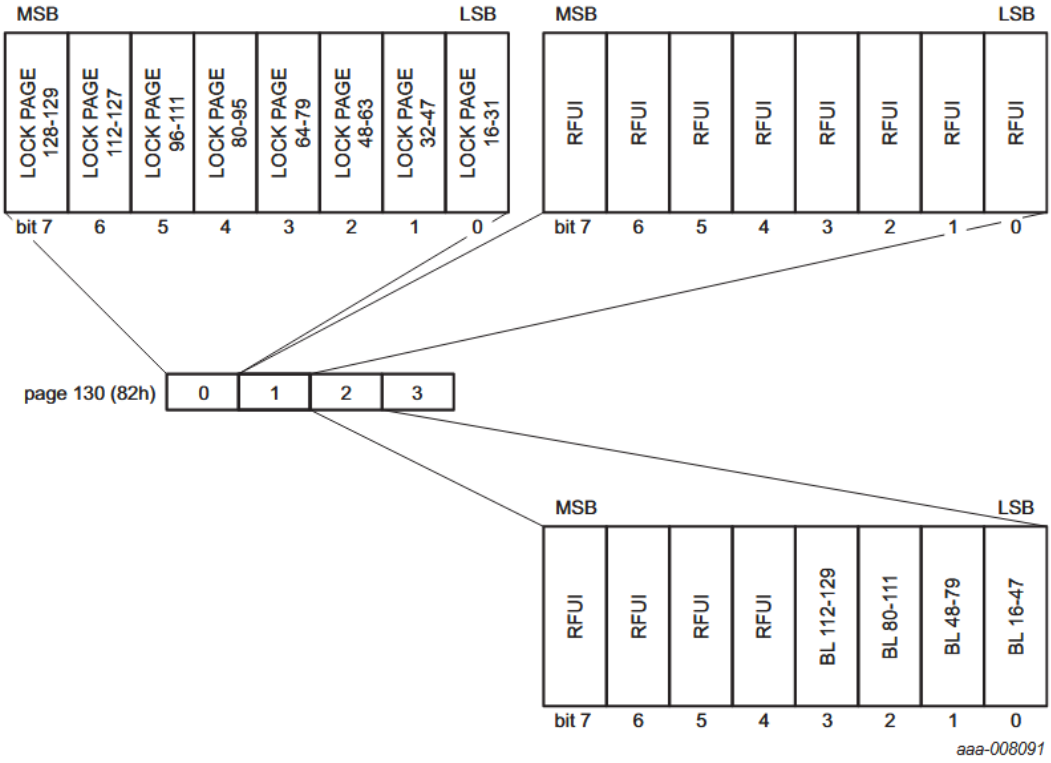


Figure 12: Diagram for dynamic locking

5.1.4.3 Using NFC Tags to Redirect to a Webpage

Our final main area of research, and arguably the field where a majority of us had the least amount of background knowledge, consisted of the use of NFC tags for redirecting to webpages. Not only was it found that performing this task was possible, but it is implemented heavily and in more complex manners. For example, NFC tags have been applied in a broad range of mobile device commands, with use increasing drastically over the past couple of years. The most recognizable use case is tap-to-pay methods found on phones and credit cards, but NFC tags have also been incorporated in the use of instant Wi-Fi and network access, automated Bluetooth pairing, or more trivial tasks like the setting of alarms. These tasks are aided through the use of apps developed that make customization accessible such as NFC Tools, which have also been increasing in availability over time.

Other uses of NFC tags that were discovered throughout our research included starting machines remotely such as cars, lights and computers through quick tap functions. The most relevant to our application, however, was found in smart posters. Smart posters consist of movie posters that have NFC tags embedded in certain parts of them. After a user taps the poster, users are immediately redirected to a URL where a trailer of said movie begins to play. The discovery of this technology proved to be highly valuable as it let our group know that NFC tags could be used to redirect to webpages and introduced us to the idea of static URLS. Which, in short, is the process of embedding a static URL on an NFC tag and simply redirecting that URL to a desired page. This concept is planned to be used in our current NFC solution while other actions are still being considered. These actions include students directly connecting to the routers network through the tapping of a tag or user use metrics being maintained as a tag is tapped by different devices.

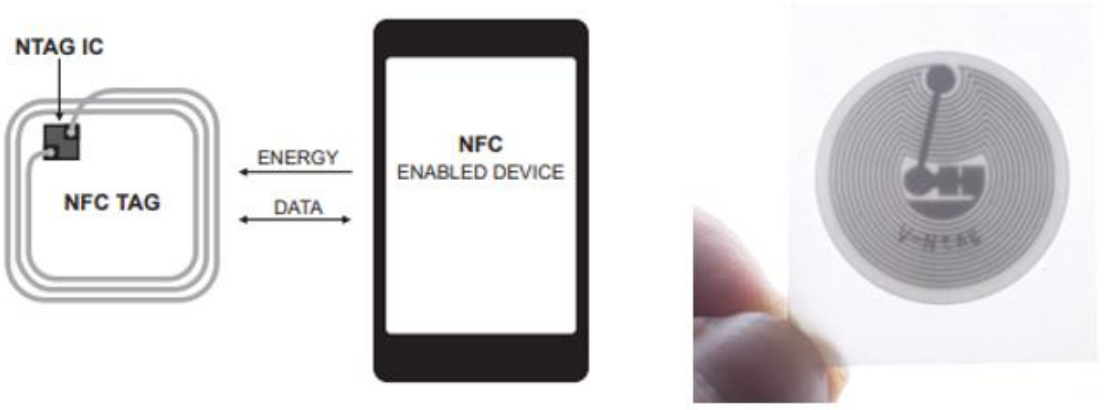


Figure 13: From left to right, Diagram of NFC Functionality from NTAG 215 Manual and Actual photo of NTAG 215 NFC Tag

5.2. Other Website Builders

One of the major tasks assigned to our group was the process of creating a website where users could make their own custom tour sites of any location. Using this as a basis, this became the group's first point of research, and an ample amount of time was dedicated to this topic. We sought to study well-recognized websites that allow the creation of custom websites, more specifically their process and the options they provide users. This led to our group focusing on the sites wix.com and squarespace.com. Both sites are popular website creators and provided ample insight on our own design process. Primarily focused on basic text and image insertion, as well as click and drag options, our group found that both sites provided the basic tools required to format basic websites.

Once these features were observed, the next point of interest was analyzed, the access to custom templates and tools. This portion was discovered to be very significant as the templates given were already preformatted and provided users the ability to simply replace key text and images. The custom tools also proved to be notable as they provided options for tracking features related to website statistics as well as user interactions.

Our research into online website creators played a significant role in the final decisions related to our web-builder pages' functionality. Beginning with what users would be able to achieve on their pages, we decided that the functionalities we would adopt would consist of inserting text, images, and videos. It was determined that these options would be sufficient to achieve the goal of the project while also permitting the users to have an ample amount of customizability in what they create. The main feature that we decided to implement in our design however, consisted of providing users with customizable templates for site pages. We decided this option would be the best alternative as it limits the possibility of unexpected errors arising during subsequent steps while simultaneously being an achievable goal.

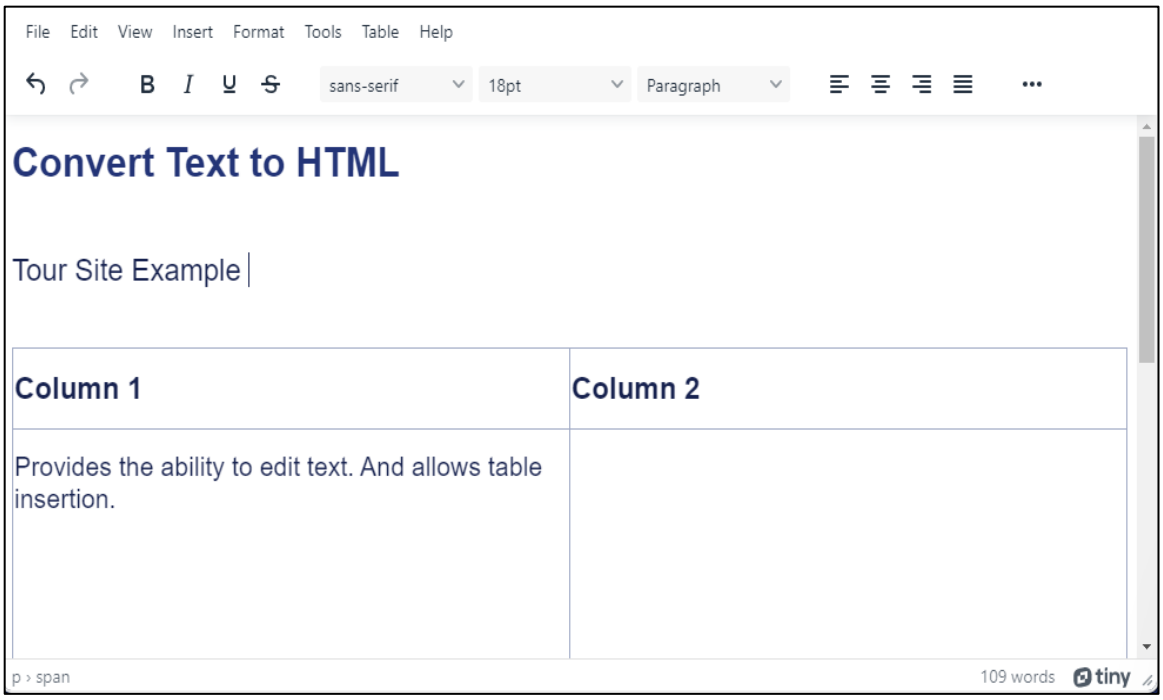


Figure 14: Image of TinyMCE Text Editor for Converting Code to HTML. Example text was inserted by the group, as well as the table.

5.3. Database

The sponsors of the project listed as one of the requirements that EMURR should have a database to centralize the digital assets from which the professors, chaperones, and tour guides, in general, can pull information to create custom tour packages websites. In addition, the sponsors suggested pairing with MySQL database, one of the most popular Open-Source SQL database management systems owned by Oracle Corporation. The choice of MySQL was made based on the fact that the Center for Humanities and Digital Research (CHDR) at UCF set up shell accounts for our team members on its Linux server running Ubuntu, and Ubuntu provides two database servers: MySQL and PostgreSQL. With that said, the database can be hosted on the CHDR server, and the project will not incur any costs associated with hosting the database with a third-party company.

Even though the terms database and database management systems (DBMS) are often used interchangeably, they defined distinct concepts. A **database** is a logically organized collection of information or data that is typically stored in a computer. A **DBMS** is a computer program that allows users to manage the database. Together, they are referred to as a database system which is usually shortened to simply database.

Our sponsors are flexible in terms of selecting the software stack to support the development of the tour builder website. Despite the requirements and the possibility of hosting the database in the CHDR server, one of the first steps taken towards constructing the database was doing some research on the different options of database systems. The research was to determine if the sponsors' recommendations fit the project needs and what other alternatives are available that could accommodate these requirements better than the suggested approach of using MySQL.

During the early stages of designing a new project or application, discussion of database requirements, types, and implementation will often come up, and it is not always an easy choice for the team. The first major decision to be made is whether to opt for a relational database or a non-relational database. Therefore, it is important for the team to understand the difference between these two types of databases to narrow down the list of potential database candidates. One type of database is not necessarily better than the other, but each has contrasting reasons and use-cases that make them stand out in particular areas. This section will discuss the key features of these two types of databases, their pros and cons, and why we ultimately sided with our sponsors' requirement of using a MySQL database.

5.3.1 Relational Databases

Relational databases might be the most intuitive type that comes to our minds when talking about databases since the data is arranged nicely into rows and columns similar to an Excel sheet. A relational database is a collection of data items organized as a set of tables with columns and rows. These tables have pre-defined relationships between them. This is where the term "relational" gets its name from. Each column in the table defines the information being stored and the rows hold the actual data. On each table, there must be a column in which the rows are marked with a unique identifier called a *primary key (PK)*. The primary key can be used in other tables to establish relationships among multiple tables in the database. When a primary key of one table is used in another table, then this column on the second table is known as a *foreign key (FK).*

Most of the time, primary keys are associated with an auto-increment field that allows a unique index to be generated automatically when a new record/row is added to the table. To illustrate these concepts, consider the following relationship example between records contained in two tables as shown in Figure 6.

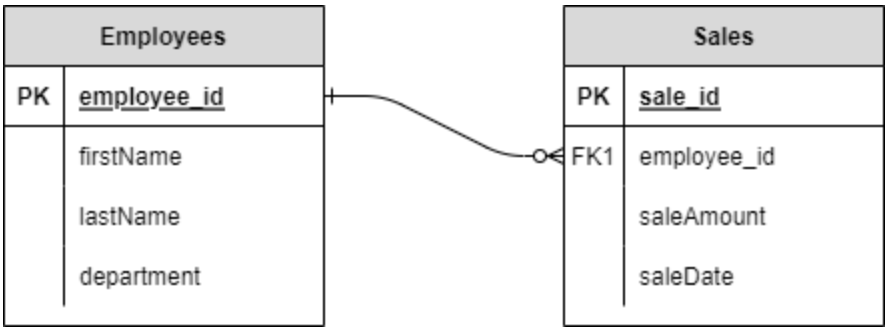


Figure 15: Entity Relation Diagram (ERD).

Each row in the Employees table has a unique id (primary key) starting at one for each new employee inserted in the table. The Sales table contains sales records associated with the employee that made the sale. The “employee_id” is the foreign key on the Sales table, and it can appear several times meaning the same employee made more than one sale.

The most common way to store, manipulate, and retrieve data in a database is using SQL (Structured Query Language). Relational databases are also referred to as SQL databases. SQL is a standardized programming language to write SQL queries to perform CRUD (Create, Read, Update, Delete) operations in the database. SQL became a standard of the American National Standards Institute (ANSI) in 1986 and it is supported by all major relational databases. A simple query example to retrieve all the sales made by an employee whose id is 3 in Figure 2 is as follows:

```
SELECT * FROM Sales WHERE employee_id = 3;
```

Another relevant factor of relational databases is the schema. The process of designing the schema or how the data is organized in the database is known as data modeling. The schema is a visual diagram that shows the table titles, fields/columns, data types, primary and foreign keys, and the relationships between tables. These relationships can be of the type one-to-many (most common), many-to-many, or one-to-one.

In a one-to-many relationship, a row in table A can have multiple matching rows in table B, and a row in table B can only have one matching row in table A. Figure 2 represents this type of relationship. On the other hand, in a many-to-many relationship, a row in table A can have multiple matching rows in table B and vice-versa. Lastly, a one-to-one relationship is not very common, but it occurs when a row in table A can have only one matching row in table B and vice-versa [14].

According to Statista [15], a German company specializing in marker and consumer data, as of January 2022, the most popular relational database management systems (RDBMS) worldwide are listed below (in popularity order):

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL

To end the subject of relational databases below is a summary of their key features, pros, and cons:

Key Features:

- The data is stored in tables.
- The data in a column/field is accessible by specifying the table name, the column name, and the primary key of the row.
- A schema defines the relationship between tables and field types.
- Table rows have a unique identifier called a primary key that when used in another table is called a foreign key.

Pros:

- Atomicity, Consistency, Isolation, and Durability (ACID) compliance
    - ACID is a set of properties that guarantees the reliability of the transactions in a database. The general norm is if one operation fails, the entire transaction fails, and the database will remain unchanged as it was before the transition was triggered. Below is a brief description of each term.
    - **Atomicity:** all operations from one transaction must be successfully executed; otherwise, the entire transaction is invalidated. This means that the data being modified cannot be accessed until all the operations are completed. Atomicity prevents querying wrong data that may occur when the data has been partially updated as there are still operations to be executed.

    - **Consistency:** refers to the idea that all data written to the database in a transaction must adhere to all pre-established rules and restrictions.
    - **Isolation:** the main goal is achieving concurrency control and ensuring each transaction is independent of other transactions. Isolation guarantees that the transactions executing at the same time leave the database in the same state that it would be if all the transactions were to be executed sequentially.
    - **Durability:** entails that all the changes made to the database are permanent once a transaction is successfully executed. The data will remain unchanged until another transaction modifies it [16].
- Data integrity as a measurement of accuracy and consistency of data
    - Primary and foreign keys ensure there is no duplicated data
- Simplicity
    - Relational databases have been around for quite a long now and there are multiple software programs and resources to help manage the database. Additionally, the syntax of the SQL language is intuitive to understand even for non-programmers to generate database queries

Cons:

- Flexibility
    - Generally, the schema of RDMS is rigid meaning the columns and datatypes of the columns (including restraints in format and length) have been defined beforehand. Therefore, making changes to the schema is complex because it requires the database to be offline while changing all the data. If all the necessary data fields for an application are not well-defined at the start, a RDMS may not be the best option.
- Performance
    - It is related to the complexity of the database schema: the number of tables and rows on each table. The time to perform queries will increase as the complexity of the database escalates.

5.3.2 Non-Relational Databases

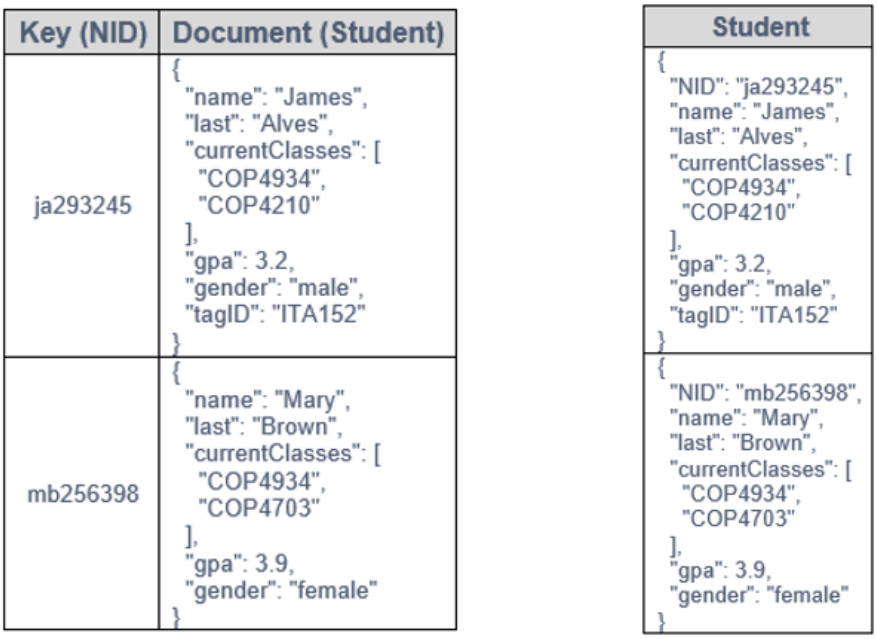
A non-relational database is also known as NoSQL which stands for “Not Only SQL.” Although non-relational databases support SQL-compatible queries, typically the store mechanism uses other programming languages to query the data. By definition, a non-relational database is any database that does not use a rigid schema of tables, rows, and columns to store data. Rather than trying to fit the data into a tabular schema, its storage model is optimized for the specific requirements of the type of data being stored [17].

Non-relational databases have been developed with cloud computing in mind because unlike relational databases, they allow for flexibility and scalability for storing and accessing unstructured data. This is a great advantage over relational databases when all the data to be stored cannot be identified at the early stage of the application or is initially unknown.

Depending on the way of storing the data, non-relational databases are broken down into four types:

- Document-Oriented Database or Document Store.
    - As the name suggests, the data is stored in documents that usually follow a JSON-like format. This type of database pairs a key with a much more complex data structure (document) like key/values pair but in this case, the values correspond to a document. As shown in Figure 7, left, there is no fixed schema, and the document object provides great flexibility should additional attributes be added to it.
- Key-Value Stores.
    - The most basic type among no-relational databases where each key is associated with only one value in the collection. (See Figure 7, right).
- Colum-Oriented Database.
    - It uses tables, rows, and columns to store data, but the names and data types of the columns may vary from row to row in the same table. That is, the structure of the data is not rigid as in relational databases.
- Graph Stores.

- The most specialized of the four types. In a graph database, the nodes store the data and the edges between them represent the attributes of their relationship.



Key (NID)	Document (Student)
ja293245	{ "name": "James", "last": "Alves", "currentClasses": ["COP4934", "COP4210"], "gpa": 3.2, "gender": "male", "tagID": "ITA152" }
mb256398	{ "name": "Mary", "last": "Brown", "currentClasses": ["COP4934", "COP4703"], "gpa": 3.9, "gender": "female" }

Student
{ "NID": "ja293245", "name": "James", "last": "Alves", "currentClasses": ["COP4934", "COP4210"], "gpa": 3.2, "gender": "male", "tagID": "ITA152" }
{ "NID": "mb256398", "name": "Mary", "last": "Brown", "currentClasses": ["COP4934", "COP4703"], "gpa": 3.9, "gender": "female" }

Figure 16: From left to right, Document-Oriented DB and Key-Value DB

As per Google trends, the top NoSQL database engines are (in popularity order):

- MongoDB:
  - This is currently the most popular and uses JSON-like documents to store data. MongoDB also allows auto-sharding which is a method for partitioning very large databases or high throughput applications into smaller, faster, and more manageable sets distributed across multiple servers [18].
- Cassandra:
  - This is a column-oriented no-relational database
- Redis (Remote Dictionary Server)
  - It is a key-value no-relational database with the drawback that requires knowledge of Lua, a high-level programming language.

To wrap up non-relational databases below is a summary of the key features, as well as the pros and cons associated with them:

Key Features:

- Support for various data models/structures: document-oriented; key-value stores; column-oriented; graph stores.
- The data does not need to have a defined scheme upfront to be stored.
- The storage model (one of four) is best suited for the type of data being stored
- Support different types of abstract data structures: lists, maps, sets.

Pros:

- Schema flexibility: the data does not need to have a predefined structure. Perhaps, it can be stored in a more free-form format.
- Can handle any data format such as structured, semi-structured, and unstructured data.
- Horizontal scalability: designed for the cloud and when the load increases, NoSQL databases can scale out by using clusters of hardware (multiple servers acting as a single system) instead of scaling up (vertical scalability) by adding robust servers.
- High performance and availability are achieved given that the data is replicated across multiple cloud resources (data centers and servers) minimizing latency for users.

Cons:

- Data consistency and integrity. To achieve flexible data models, NoSQL databases relax some of the data consistency rules and restrictions that are crucial in relational databases.
- No standardized language as SQL is for relational databases. The syntax to query the data varies for the different types of databases.
- Not good for complex queries due to the variety of data structures of the NoSQL databases.

5.3.3 Database Type Selection

After researching the key aspects of both relational and non-relational databases and pondering their pros and cons, our team concluded that a relational database would be more appealing for the project. This type of database excels when the schema is predictable during the data modeling which implies knowing the structure, tables, fields, and the relationships between tables upfront. Besides, the relationship between the tables is important in the model of our application. We will delve into this topic in more detail in the Database Design section. Our team has met with the sponsors multiple times, and we have put together an exhaustive list of all the necessary data for the tour builder website.

Some of the aspects considered before choosing the database were:

- The type of data we will be storing
    - Our data fit well into a rows and columns layout similar to a spreadsheet.
- The size of the data
    - A rule of thumb experts agrees upon is that the bigger the data set, the more likely a NoSQL database will be a better option, but our data set is small. The use-case given by the sponsor is the database will store initially ~200 biographies of veterans buried across 5 cemeteries in Central Florida.
- Dedicated resources to support and maintain the application after the project is released to the UCF CHDR.
    - Typically, a SQL database requires less time to manage and expertise whereas a NoSQL database requires more programming knowledge. Also, SQL is an English-like language. Writing SQL queries is like writing English sentences which makes it more manageable.

Once a SQL database type was selected, the next step is choosing the RDBMS to interact with the data. As mentioned before, our database will be hosted on the CHDR Ubuntu Server which supports two popular SQL database servers: MySQL and PostgreSQL. The next two sections will go over the features and use-cases of both databases to help us decide which option is best suited for the application.

5.3.4 MySQL Overview

MySQL is the world’s most popular open-source RDBMS. As stated previously, the data in a relational database is stored in tabular form rather than in a big document structure. The “SQL” part of “MySQL” means the standardized language is used to access the database by either entering SQL statements directly in the database or embedding them into code written in a different language.

The most popular use-case of MySQL is web applications. It is impossible to go far on the web without running into an application supported by this database. In fact, most developers agreed that MySQL is better for building websites and applications where speed is a big concern, e.g., online transactions because is light on features compared to PostgreSQL. Among the major companies using MySQL are NASA, BoA, Netflix, Facebook, Twitter, etc.

MySQL is simpler and relatively easy to set up and manage; as such, it can run well with minimal oversight by developers. Moreover, it has great community support and online resources more than any other database not only for being open source but also the most popular RDBMS out there.

5.3.5 PostgreSQL Overview

PostgreSQL is the world's most advanced open-source RDBMS. It is an object-relational database whereas MySQL is a purely relational database. This feature is representative of a key differentiator between both databases.

PostgreSQL is popular for building more complex websites and applications that require highly customizable database solutions. Generally, PostgreSQL works best for large and complicated analytical processes as it comes with a slew of great features such as table inheritance, function overloading, extensibility, and native NoSQL capabilities to handle complex queries and massive databases. Companies tend to have dedicated database technicians to perform admin tasks on the database and create and implement custom functionalities. Some of the global giant tech corporations using PostgreSQL are Apple, Cisco, IMDB, and Instagram.

Even though PostgreSQL has fewer support resources than MySQL, the fact of being an open-source software makes it possible to find a wide variety of tutorials and recommendations on its official website and forums. Because of the high customization permitted by PostgreSQL, the main concern when running into a niche use case or specific issue is that there may not always be a resource directly related to the particular problem. Therefore, IT professionals need to do more work on their own to troubleshoot the issue.

5.3.6 Why MySQL?

Based on the preliminary research conducted, both MySQL and PostgreSQL are highly capable database tools for developing the backend of our application which makes it difficult to choose between the two.

We ultimately sided with the MySQL engine for the following reasons:

- It is preferable for web applications –like our tour builder website- that do not require complex queries or any of the advanced features offered by PostgreSQL as explained in the previous section.
- It is overall a simpler tool that requires less expertise and database knowledge to set up, manage, and oversight.
- The small size of the data our application needs to handle. For the preliminary design of the database schema, we are contemplating only two tables: users and users' posts. However, this is subject to change during the implementation.
- The scope of the application is limited to minority and underfunded organizations and community groups. It is not expected to grow up to the enterprise level.
- It can be hosted for free in the CHDR and we will not have to search for a cloud server hosting provider. Additionally, we will meet the sponsors' requirements of using a MySQL database.
- It has more community support, online resources, and third-party tools since it is more widely used and popular than PostgreSQL. Even though we are not foreseeing running into any issues.

5.3.7 Database Management Software

Technically, a Database Management System (DBMS) is a database management software responsible for performing all sets of administrative tasks in the data like creating, updating, retrieving, or querying the data in general, managing user permissions, and monitoring the status of the database. DBMSs usually have a graphical unit interface (GUI) to facilitate such tasks. Due to the fact that MySQL is among the most popular and widely used databases, there exists a lot of DBMSs supporting this type of database, and it can be difficult to choose a tool that best fits our requirements. The next two sections take a closer look at two of these tools.

5.3.7.1 phpMyAdmin

phpMyAdmin is one of the most popular, free, and open-source administration tools available to manage MySQL databases over the web. It is written in PHP and has a very intuitive user interface. It supports a variety of functions such as managing the database and its tables as well as indexes and user permissions. It has a query interface to type and run SQL commands. Due to its free nature, phpMyAdmin has a lot of community support and learning materials. It is currently hosted on GitHub. Some of its key features are listed below:

- Friendly and easy to use web-based user interface. This is probably the main advantage of using phpMyAdmin over Workbench client.
- Supports most of the operations in a MySQL database like managing user accounts and privileges; execute/edit/bookmark SQL statements; create/delete/rename/copy and browse databases, tables, fields, and indexes.
- Import and export data into multiple formats (CVS, SQL files, etc.)
- Global search capability in the database.

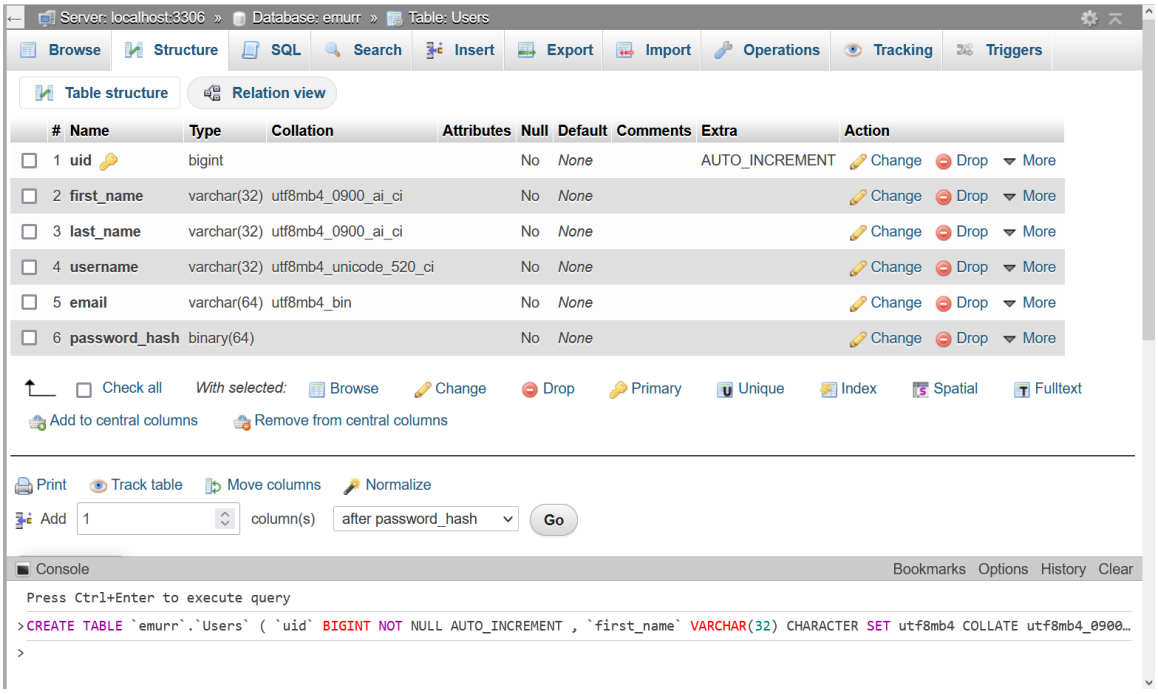


Figure 17: Example of a newly created “Users” table in phpMyAdmin.

5.3.7.2 MySQL Workbench

MySQL Workbench is also a tool to manage MySQL databases. It is a desktop application that can be installed on Windows, Linux, and Mac OS X. One of the features making the tool stand out over phpMyAdmin is the data modeling which lets users visually design the database schema and then export it as a SQL code to generate the database.

This process of transforming a data model into a physical database is referred to as "forward engineering". The forward engineering wizard generates the SQL code automatically contributing to eliminating the error-prone process of manually writing the SQL code. Similarly, the "reverse engineering" wizard lets you get a better insight into the database schema [19].

Workbench is a more comprehensive tool than phpMyAdmin for database management. It provides an enhanced SQL editor with autocomplete function, color syntax-highlighting, and reuse of SQL code snippets. It has a visual performance dashboard that is a great tool to monitor the performance of MySQL databases showing a glance at the Network Status, MySQL Status, and InnoDB Status.

MySQL Workbench is available in three editions listed below. The free open-source version offers a basic set of features and the remaining two commercial editions have extended functionalities like schema and model validation.

- MySQL Workbench Community Edition - Open Source (GPL License)
- MySQL Workbench Standard Edition - Commercial
- MySQL Workbench Enterprise Edition - Commercial

Both tools, phpMyAdmin and the free edition of MySQL Workbench, are suited for managing the database of our application, and we have decided to use both interchangeably. For example, there may be some instances where the team is working in the SD Lab and phpMyAdmin will be best suited because it can be easily accessed from the web browser of any computer. Workbench on the contrary requires a one-time installation of the application in the operating system of our choice. This can be appropriated to manage the database from our personal computers, and this way take advantage of the extra features this tool offers. The ERD of the database in section 7 of the document will be created with the help of the Model feature of MySQL Workbench.

5.4. Hosting Services

Hosting a website means make the website accessible via the World Wide Web (WWW) by providing the hardware and software infrastructure that the website needs to operate. There are two ways to host a website: use a hosting provider or host locally on a computer.

The website can be hosted in a local machine running either Windows or Linux operating systems and installing the required software programs to run the server. Then it is necessary to make the server reachable from the Internet. The process can be cumbersome as requires setting up the home router to receive request on its public IP address assigned by the Internet Service Provider (ISP) and forward the request to the dedicated computer in the local network. After this step, if everything was configured correctly, when enter the public IP address on the browser the website should be visible from the outside world. In addition, if we want to give the website a name for easy access like *yourwebsitename.com*, we could buy a domain and point the domain to the public IP address.

The hardware component for running the server is another important element to take into consideration while self-hosting. Having a reliable server to handle volume of visitors requires high-end equipment (processor, memory, and storage) and redundancy in hardware. Also, the network infrastructure (routers and modems) should be able to handle high traffic loads.

In summary, setting up a local server could be a good learning exercise, but it requires time investment and a more hands-on approach. This method has quite a few disadvantages and it is not recommended for the type of project we will be developing. Most of the time is less reliable than hosting the server with a cloud service provider which are platforms that take care of the above-mentioned details allowing you to focus only on building a quality website.

5.4.1 UCF CHDR Server

Our sponsors set up shell accounts for each team member and a MySQL database for the EMURR project in an Ubuntu server owned and maintained by the Center for Humanities and Digital Research (CHDR) at UCF. As the project is handed off to the sponsors, the plan is to host all parts of the web builder application (Next.js application and database) on the CHDR server.

During the development of the application, our team is proposing a hybrid approach of meeting in the Senior Design Lab once a week and the rest of the time remote, meaning it will be necessary to sign onto UCF Virtual Private Network (VPN) first before accessing any UCF campus resources from an off-campus location. This will add an extra step every time we try to access the CHDR server for developing.

Moreover, during the developing stage of the project, it would be required to install and update dependencies and packages as we go but given that we do not have admin rights on the CHDR server, we must contact the server administrator, Connie Harper, when running into permission problems which might cause delays in the project.

These drawbacks will slow down the development process, and we as a team have agreed upon using a cloud service provider to host the application as an interim solution to the said problems. The cloud offers unique features to handle deployments with a few clicks and a user-friendly interface to manage all aspects of the hosting such as automatic integration with the source control repository which facilitates the process of promoting code from the development environment to production and team collaboration. Source control integration also allows to roll back easily to previous versions. Using a service provider will shield us from the technical aspects of the hosting and we can concentrate our efforts on building the application. Another aspect to consider is that generally, these service providers offer 24/7 technical support. Whereas the CHDR server does not offer these facilities. Just the fact of pushing the changes or updates in the source code manually to the server is time-consuming.

However, once the web application is completed and ready to be handed off to the sponsors, we would schedule a meeting with Connie to deploy it onto the CHDR server in the case elevated privileges are needed to install certain packages. We have researched how to deploy a Next.js project on an Ubuntu server with NGINX but since this step will be completed towards the end of the Fall 2022 semester, we do not have specific details of the process at present time. We will be updating this document accordingly with the details of deployment onto the UCF server.

5.4.2 Virtual Machine in Senior Design Lab

Since the team will be meeting weekly in the Senior Design Lab, it was proposed to set up a virtual machine in the dedicated server running Apache to have a testing environment similar to the CHDR server where the final application will be deployed. In this manner, we will ensure a smooth transition once the final product is ready for deployment onto the CHDR server and at the same time, we would have more control and freedom to make any changes in the VM server.

We will be using the VM in parallel with the cloud service provider chosen for the developing stage of the application. Consequently, the following sections present an overview of hosting providers considered for the application. We would like to host the database and the website with the same provider and free of cost.

5.4.3 Microsoft Azure

Azure is the could computing hosting service operated by Microsoft. Its closest competitors are AWS and Google Cloud Platform. If you are a new customer and have not had an Azure account in the past, you can create a free account and receivere$200 in credit to use in the first 30 days. Even though it is a free account, the sign-in process requires identity verification by a credit card, but you will not be charged unless you move to a pay-as-you-go pricing. With the newly created account, Azure offers popular services free for 12 months and 40+ services always free.

The free account allows to try Azure Database for MySQL-Flexible server to manage MySQL databases in the cloud with a monthly limit of up to 32GB of storage and the App Service for web hosting has a free tier option for smaller websites with 1 GB of storage space, a "*.azurewebsites.net"* subdomain and the option to automate the workflow to deploy the deploy from a GitHub repository. There is a price calculator to estimate how much hosting on the platform will cost depending on the services required.

What makes Azure shine among its rivals are features like backups, scaling, and disaster recovery which are must-haves for big corporations and website handling high loads of traffic, but they are out of scope for the application we are building.

At the time of research, Azure also has a plan for students like the one for new uses. All we have to do is use the university email to sign up. The initial credit is $100 and no credit card is required for identity verification. The fully managed Azure Database for MySQL is also free for 12 months and the basic tier of the App Service is always free.

If during the development of the application, we were to upgrade from the free tier of the App Service, the pay-as-you-go pricing of Azure offers flexibility as we would only pay for the services used and cancel anytime, but rates are higher which leaves them out as one of our possible options. Furthermore, nobody in the team has prior experience with Azure.

5.4.4 Amazon Web Services (AWS)

AWS is the world's most popular, comprehensive, and adopted could computing platform that offers over 200+ fully featured services -more than any other cloud provider- from data centers worldwide [20]. These services can be classified as follows:

- Infrastructure Technologies: compute, storage, and databases

- Emerging Technologies: machine learning, artificial intelligence, data lakes and analytics, and IoT.

One of the features that makes AWS the leading provider of cloud services is that no other competitor offers as many Regions[2] with Availably Zones[3]. At the time this document is being written, AWS has 84 Availability Zones and 26 Regions globally and has announced an additional 24 Availability Zones and 8 Regions.

AWS has wide range of solutions for various end uses in terms of web hosting. Some of the most popular are listed below. We will focus specifically on the products needed to deploy our project.

- Static Hosting with S3
- Amazon Lightsail to launch a WordPress site.
- AWS Amplify
- AWS Elastic Beanstalk
- Amazon Elastic Compute Cloud (Amazon EC2) or Amazon Elastic Container Service (Amazon ECS)

5.4.4.1 Amazon EC2

Amazon EC2 allows obtaining virtual servers or computer instances in the cloud quickly and inexpensively with a few clicks in the AWS Management Console or via an API using an SDK. It is one of the oldest services and the lowest level building block of AWS. The EC2 instance will be running within minutes, and we would have access with full administrative control which is one of the reasons we are using a cloud services provider in lieu of the CHDR server for development.

EC2 is a type of service called Infrastructure as a Service (IaaS) also known as cloud infrastructure services. IaaS offers essential storage, compute, and networking resources on demand to the end users. Before deploying an application to an EC2 instance, it is necessary to create a virtual server based on the needs of the project. In other words, the team will need to manually configure the virtual server, database instance, and related software to launch the application.

---

[2] A Region is a physical location in the world that consists of 2 or more availability zones.
[3] An Availability Zone is one or more discrete data centers, each with redundant power, networking, and connectivity housed in separate facilities.

EC2 provides a range of instances for different use cases from small and economic instances best suited for low-volume applications up to cluster compute instances designed for high-performance computing workloads. The instances are optimized for compute, storage, memory, and GPU processing allowing the users to find the right price-performance combination based on the workload. EC2 offers flexible price options with no upfront fees including pay-as-you-go and the instances can be scaled up or down depending on the users' needs. The free tier includes 750 hours of Linux/Windows "*t2.micro"* server instances monthly for one year.

EC2 has also several security features. The instances are in a Virtual Private Cloud (VPC) that is a logically isolated network controlled by the user and lets users choose who can access the instances.

5.4.4.2 Amazon RDS

Amazon Relational Database Service (Amazon RDS) is a tool for managing popular relational databases: MySQL, PostgreSQL, MariaDB, and SQL Server, and Amazon's own relational database built for the cloud, Aurora. Amazon Aurora is compatible with MySQL and PostgreSQL and it is ten times cheaper.

Amazon RDS allows automation of time-consuming administration tasks with a few clicks or API calls. These tasks include backups, access management, point in time restore, disaster recovery, encryption, secure networking, scalability, monitoring, performance optimization, etc.

Similar to Amazon EC2, Amazon RDS has a free tier option with 750 hours per month of database usage for one year. However, only instances running MySQL, MariaDB, and PostgreSQL engines are available in the free tier option.

5.4.5 Heroku

Heroku is also a cloud service provider platform which facilitates fast and effective building, deploying, monitoring, and scaling of web applications. Heroku is hosted on Amazon's data centers, and it offers a type of service called Platform as a Service (PaaS) where the hardware, software and infrastructure are delivered to the users as complete cloud platform. Apps built on Heroku run in smart containers called dynos. Heroku supports over 200+ addons and is more user friendly compared to AWS to deploy and scale applications.

The main advantage of using Heroku is that all team members are familiar with it from past project assignments. Its integration with GitHub to deploy applications is pretty straightforward. All you need to do is link the GitHub account to Heroku and choose the repository of the application. Once the repository is connected you can either deploy a specific GitHub branch or enable automatic deployments from GitHub in which every new push to a specific branch will deploy a new version of the application.

One disadvantage to Heroku is the hourly prices are twice as much as the AWS after scaling up from the free tier option. Heroku was initially our first option for hosting the application. Matter of fact, the application was initially deployed on Heroku.

5.4.6 AWS vs Heroku

Features	AWS EC2	Heroku
Owned by	Amazon	Salesforce
Hosted on	Proprietary services	Amazon's data centers
Service Type	IaaS	PaaS
Deployment	More difficult and complex	Rapid, ready-to-use
Pricing (basic)	1 year free	Free
Pricing (similar instances)	*t3.micro* (1GiB RAM) $7.48 per month	*standard-2x dyno* (1GB RAM) $50.00 per month
Ease of use	From medium to advance	For beginners
Complexity	Complex, Recommended a DevOps Engineer	Simple, great for beginners
Management Tools	- AWS CLI - AWS Management Console - SSH	- Heroku CLI - Heroku Connect (add-on) - Heroku Status

Table 3: Comparison Table. Major differences between Amazon EC2 and Heroku.

5.4.7 Our choice (AWS)

Our application can be deployed using the free tier options of both Heroku and AWS. Even though our team is familiar with Heroku from past school projects, and we went through the exercise of setting up a ClearDB database and deploying code to Heroku from a GitHub repository which is easier than AWS, the team sided with AWS as a hosting provider. The main reason for our selection is because AWS is the leading cloud platform nowadays and cloud-based skills are in high demand in the job market, and the team is aware that having the opportunity to work with AWS would be one of our major takeaways from the senior design project. Moreover, some of us have expressed interest in obtaining some AWS certifications and being able to familiarize ourselves with Amazon RDS and Amazon EC2 will most likely benefit us in the exam and the real world in general.

5.5. Raspberry Pi & Router Hosting

5.5.1 Using A Router to Host a Local Raspberry Pi Website

The need to maintain a connection between our Raspberry Pi hosted website and router, ultimately led to the next point of research, existing methods for hosting a website with router implementation. Primarily revolved around confirming its feasibility and position as the best alternative, finding varying methods of application was the group's main goal during this process. Fortunately, hosting sites on Raspberry Pi's is a standard practice, and there was no shortage of the resources available describing how individuals can implement the task and how to troubleshoot potential errors. It was also discovered that a standard order is followed for most implementations, with differences consisting mostly of settings related to personal preference.

A summarized version of the standard practice is as follows. The Raspberry Pi is first connected to a router preferably through an ethernet cable. Doing this provides a standard SSH connection. Trivial configurations are then performed on the Pi before most other connections are formed. This may include updating the pi, changing layouts, and setting other user features. Afterwards a user would install their preferred server. Examples may include Apache and MySQL to name a few. Following this, Port Triggers are set up to allow external communication to the Pi. Examples of this communication may include individuals solely trying to access the site on the Pi using HTTP or permitting the use of other ports. Finally, IP configurations and domain related information are changed for easier access to the Raspberry Pi's website. Fortunately, our process for configuring the connection between the Pi and the router was similar to the standard method, so implementing changes in the procedure was not necessary.

5.5.2 Raspberry Pi Hosting Prototype

Once the Raspberry Pi is holding sites in memory it must have the capability to host them for users. The process for hosting sites is not set in stone but some initial proof of concept testing has been completed on a Raspberry Pi. The following is an exploration into how we decided to attempt Raspberry Pi site hosting.

We begin the proof of concept by flashing the Operating System (OS) onto the Raspberry Pi. In the final product the operating system will be a headless version of the regular Raspberry Pi OS, also known as Raspbian.

However, for proof of concept we used the regular Raspberry Pi OS. The only difference between a regular OS and headless OS is that the regular OS has a Graphical User Interface (GUI) without any loss of generality or functionality. This small difference in the OS helps us to develop and test with further clarity by using intuitive visual aids. Flashing the OS onto the Raspberry Pi first required downloading the Raspberry Pi Imager from the manufacturer's official site. After downloading the disk imager, we merely plug in a micro SD card to flash the OS onto. Finally, when that is complete it is simply a matter of putting the SD card into the Pi and now whenever it turns on it will boot the proper OS.

The next major task in our proof of concept is to host a live server using Visual Studio (VS) Code and the live server extension. The main benefit of using VS Code and the live server extension is that we can edit pages and host them all in one singular space. This reduces friction between editing and visual feedback. Getting VS Code on the Pi is as simple as opening the terminal and running the command `sudo apt install code`. We understand there are more elegant ways of achieving this goal but for testing this methodology has been sufficient.

Once we are finished installing VS Code, we can create a basic HTML page. We start with some basic HTML and some contents to get started, adding any CSS or JavaScript to our liking. Now that our page is written we go to the extension's menu of VS Code and search for the live server extension and add it. Once the live server extension is installed running the server is just a matter of clicking the button generated by the extension at the bottom of VS Code or following the built-in keyboard shortcut. With the live server extension active the server should now be hosted on the same Wi-Fi network as the Pi. In order to connect to the site once it is hosted, we will run in the terminal of our Pi `ifconfig` which will show us its IP address. Any other device on the same Wi-Fi as the Pi can use this IP followed by the port number associated with the live server in a web browser to connect.

A note on types of Wi-Fi setups: enterprise networks such as UCF Guest, UCF WPA2 Wi-Fi, and eduroam require extra care and manipulation in order to cooperate with the Raspberry Pi. The main problem motivating this aside is that when trying to connect to these Wi-Fi networks the Pi will simply refuse to recognize them. As we learned, the issue is caused by the default network manager in the Raspberry Pi OS. This networking default interferes with many enterprise networks and typical configurations such as eduroam. The simple solution is to simply use an ethernet cord, for most testing and the intended future development, this is the primary solution. This may be achievable once we have a fully functioning hardware stack as we will simply ethernet the Pi into our own router. However, at the time of writing this has not been possible. Even with a functioning Pi it is possible to host a site by ethernet in locations such as the senior design lab. However, if we are at any location on campus where ethernet is not available then further configurations have to be taken. In order to understand how to set configurations for a particular context we must first understand what the default settings are. By default the Raspberry Pi OS uses Dynamic Host Configuration Protocol (DHCP) with a respective client daemon (DHCPCD). Essentially DHCP and DHCPCD by extension are incompatible with many enterprise networks including UCF's. In order to remedy this we will use a different method of network management which is compatible with UCF. We have chosen to install the GNOME project network manager with the command `sudo apt install network-manager network-manager-gnome` from the terminal. Now that it is installed on the system we need to switch the configuration to default to the new network manager. We do this by running the command `sudo systemctl disable --now dhcpcd` to disable the default and `sudo systemctl enable --now network-manager` to enable the new network management system. Even with these steps there appears to be a bit of luck involving whether the Pi agrees with the UCF Wi-Fi, but typically a few reboots and switching settings does the trick. Another step which is not clear in its effective benefits is manually editing the WPA supplicant configuration file (~/etc/wpa_supplicant/wpa_supplicant.conf). The purpose of doing this is ensuring some network specific settings are "hard coded" so to speak. In the JavaScript Object Notation (JSON) format this file may define a network object to have fields such as network name aka Service Set Identifier (SSID), certificates (ca_cert), key management (key_mgmt), and much more. Defining these such fields doesn't seem to have any effect on the overall performance of connection to enterprise networks but appears to be good practice, especially for moody networks.

In any event, this should not be a major worry during implementation as this will not be the intended network setup for the EMURR system. The only concern and purpose for mentioning it here is if we do any demos on UCF campus in the future. In which case we will be sure to refer back to this documentation and ensure the router is working properly for our tests by then. If not, we should at least configure the Raspberry Pi with proper networking settings appropriate for the context.

5.5.3 Future of Raspberry Pi Hosting

Moving forward, the team is committed to pursuing a more sophisticated solution to hosting a web server on the Raspberry Pi. Such solutions consider web server technology such as Apache or NGINX. We have intentionally excluded framework software such as Node JS or Next JS. The reason for the exclusion is because as the Raspberry Pi will only host static sites it does not need JavaScript runtimes or any additional functionality past simply serving pages to visiting users. Similarly, the Raspberry Pi has a constraint on how much memory it has available at any moment and therefore a web server that intentionally minimizes memory impact would be preferrable. Regardless of web server, we will need to enforce port forwarding on the router so that the Raspberry Pi will properly receive HTTP / HTTPS requests from the NFC tags. Furthermore, the Pi will need to maintain a static IP address in order to support port forwarding.

First, we explored Apache which is a web server released in 1995 which supports PHP scripting. Apache is often seen as the default for many web servers as it has been standard across the internet for decades. Despite its age, Apache accounted for a plurality of all web servers on the internet just until the beginning of 2022, overcome for the first time only by NGINX. Furthermore, because all of the EMURR team has taken, or is currently taking, Processes of Object-Oriented Programming we have had experience within the last year of creating a project including an Apache web server. Additionally, Apache can handle dynamic content by relying on PHP scripting should we ever want to make the tour sites dynamic. Despite these advantages, Apache also has some pitfalls. For example, Apache runs a new process for each request causing slowness under heavy loads and higher memory consumption overall. Regardless of the doors that PHP scripting opens up for the team it is often regarded as an unintuitive and frustrating language to work with. Furthermore, introducing new languages into the mix may generate confusion and distance in the stack between roles.

NGINX, the recent and popular successor to Apache, has also caught the team's interest as a web server. The main benefit of NGINX is that it is not only faster but consumes less memory than Apache, especially when delivering static webpages. This is favorable considering static web pages are the main use case of our web server and our main constraint is memory on the Raspberry Pi. Additionally, faster load times only help the user experience so that is a positive aspect worth considering. Because of this better performance NGINX is able to scale to many connections. In fact, its design was intended to support up to ten thousand simultaneous connections, known as the C10K problem. This is obviously more than necessary but knowing heavy loads will not be an issue puts some worries to rest. Another feature of NGINX that may seem overengineered for EMURR at first glance is reverse proxying. Essentially reverse proxying allows the NGINX web server to pass any requests given to it to other servers and then send back the responses they give to the user. A potential use of this is if in the future we decide to explore dynamic tour site pages we can reverse proxy to a server with better support for dynamic pages such as Apache with PHP. This may seem awkward, but it is in actuality not uncommon to do. In this sense, developing a web server on the Pi with NGINX still leaves Apache open as an option should we need it. The only foreseeable downside to NGINX is the lack of experience the group has with it or its syntax.

In conclusion, NGINX seems to be the better option for EMURR. Designed for low memory impact, low latency serving for static web pages is exactly what we need for tour sites loaded onto the Raspberry Pi. Despite familiarity with PHP and Apache we find it appropriate to pass on this experience for the better design choice. Should we ever come to terms with needing more functionality we trust NGINX reverse proxying will be able to smooth over whatever needs we have.

5.6. Related Work

5.6.1 Introduction

EMURR is a project that is relatively new in the computer science / software engineering space. Not many other projects take part in the micro processing world when it comes to web-server applications and hosting. This could be because of its large latency times (CITE) or because of its lack of supporting software for this innovative technology (RPI). In any case, EMURR is designed to bridge the gap between digital infrastructure and interpretive information because not many people are interested in sharing data off the grid, except for environmental related data (CITE), and other sensory data collected to monitor a system (CITE).

Regardless of the current support in the field of software architecture, we do acknowledge and respect the previous findings of preceding projects. The importance of learning off other systems and literature gives us insight to what works well, and what to avoid. However, not many projects before having attempted such ambitious lengths of a project such as EMURR, we find it important to understand and incorporate these designs in EMURR. Therefore, this section will explore the findings of previous projects and the generalities of how they may apply to this project.

5.6.2 Hardware Limitations

Hardware limitations derive from the workload on the processor. The workload on the processor in our case is generated from the HTTP and API requests from our clients, heavy response times in the case of a large webpage, and any other calculative or extraneous wait processes that will happen amongst the client-server interactions. With this in mind, we can take from previous studies to analyze the effectiveness of hardware processing and latency times to effectively simulate the corresponding expected results in our project.

5.6.2.1 Relative Equivalence of Other Devices

A previous study has shown the response times of using a mobile based web-server were relatively small with an average of 7.07 seconds amongst all three web servers hosted on an android device. While this may seem like a large amount of response time for a web server, it is important to note that they were not optimizing the web pages for the fastest load times and were instead testing the strain on each different server hosted on the Android platform. Thus, in the case of EMURR, we expect to have better connection times since our wireless connections will be more prominent and consistent on a router. Furthermore, the capacity at which the data will be transferred is expected to be quicker. Admittedly, the distance and objects that interfere with the connection between the client and the server that the tour locations may also affect the connection parameters and will thus cause a slight latency. Yet, we expect the distance between the tour attendees and the tour guides to be relatively small; and if the range needs to be bigger, the tour guides can always opt for a better router [21].

Though the router contributes to the connection speed, the processing power is also an important contributor to the response times of the server. Consequently, the Pi needs to process the data quickly and efficiently. In the case of the video load times, we need the Pi to be able to pack the bytes in a reasonable amount of time to send off to the router. Thus, the Pi could reach these hardware limitations as specified in the document. After all, the best case for the web servers averaged a 5.79 response time with a cumulative total of approximately 900MB over 220 successful sessions. While this study doesn't elaborate on the individual amount of data sent over a period of time, we can extrapolate and apply the data displayed in the study to how we will experience these workloads. For instance, if over 220 successful sessions and 900MB of data was distributed then around 4MB of data was distributed over a period of 5.79 seconds. Additionally, if we were to say the EMURR unit has a similar or worse processing power as the Android device used in this study, then we can expect around 4MB to be transferred in 6 seconds. This is okay for a simple webpage loading standard HTML, CSS, and JavaScript [21].

It is common knowledge to say that a simple web page with HTML, CSS, and JavaScript will be less than a megabyte on average. If not for most modern web pages now, it will hold true for the intended EMURR designed tour site pages. However, holding true to this analogy, the data transfer will become problematic when transferring video data, and will thus prompt us to come up with new solutions to loading videos on the tour sites. In any case, this study highlights the importance of mobile hosting and its expected results using the hardware in this project [21].

5.6.2.2 Power Management

Power management can be a huge issue when discussing video streaming and other heavy data streams. One study discusses the amount of energy consumed by a 802.11b PM WLAN under heavy traffic. They found that under heavy traffic the WLAN experiences almost double the amount of power consumption. However, under the right metrics, one can minimize the WLAN power consumption by managing the WLAN power states [22].

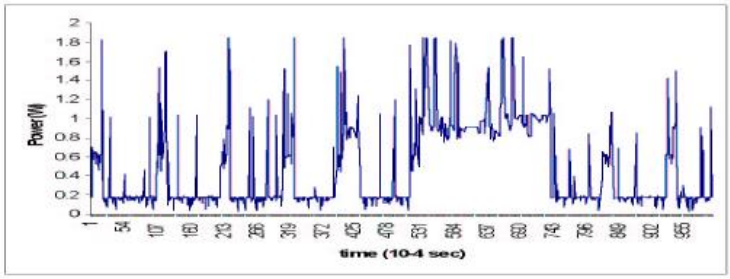


Figure 1. 802.11b PM in light traffic

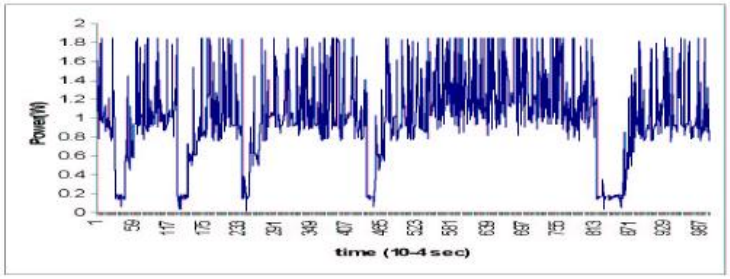


Figure 18: Power levels of WLAN (Light vs Heavy) [22]

They do this by using a Server PM that detects the amount of power consumption used by the client and server to modulate the power states of the WLAN. The Client PM communicates through a lower-bandwidth link to the Server PM to transfer this data. Using a state management system to control power levels has made an increasing difference amongst energy savings. While this is important to monitor and manage the power states for WLAN, we find that it isn't as necessary for EMURR. This is because while at tour sites, the client could make a request anytime during the tour. Therefore, we need the server and WLAN to be ready to handle the request at any point during this time. Admittedly, one could argue for making another section in the request header to alter the state of the WLAN to change the power states depending on its current status. However, this requires further research and development time to implement something like this. In this case, it would be better to implement an automatic power state management system for the WLAN as a stretch goal rather than a required feature in the final project [22].

Furthermore, it would be of note to have an option to manually change the power consumption on either the WLAN or server through a physical switch or an admin panel. Furthermore, we can possibly implement a simpler approach to power consumption by using timeouts and time ins to change the power states of the server and the WLAN. These state changes, if implemented, will be made from the server [22].

Since we are using a 12V battery to power the Pi and router, then we need to make sure we keep our power consumption within a reasonable amount to last the entire tour. It would be unfortunate for a tour guide to have made their website, upload it, and share it at the tour locations only for it to be working for 30 minutes. As a result, we can offer more encouraging solutions to remind our tour guides to charge their batteries, manage their power states, and any other simple but important ways to conserve the battery. We can do this by notifying the tour guides through the tour builder upon compilation time to the Pi to either charge their battery or to set their WLAN power state to off.

5.6.3 Environmental Implementations

Over the course of studying similar projects, most scholarly research papers deal with microcontrollers in the environment and how they deal with monitoring and sharing data over wireless connections to a central cloud server (Wireless in Agriculture). These research projects use Wireless Sensor Networks (WSN) to acquire and transfer the data over a communication network to save and distribute the results to a database. They use this data to analyze crop health, air pollution, and other impactful environmental aspects that help the human species thrive (Cite).) [23]. These research projects use Wireless Sensor Networks (WSN) to acquire and transfer the data over a communication network in order to save and distribute the results to a database. They use this data to analyze crop health, air pollution, and other impactful environmental aspects that help the human species thrive [24].

5.6.3.1 CAD Models

While these papers don't have specific relevance to EMURR, we can use the WSN to relate the effectiveness of a local network to the workflow of how the EMURR unit will work. For instance, one study discusses the importance of constructing a computer-aided design (CAD) model to design units for a robust and effective casing for their units. The CAD model houses the units which include a microcontroller, battery, soil sensors, and appropriate wiring. One aspect that is crucial to the success of the project is the weatherproof design. Similarly, EMURR will also incorporate a weatherproof design for the NFC housing CAD models as seen in the figure below.

Crucial to the success of the NFC execution, the NFC needs to be sustainable for the EMURR project to work. For example, as we build the housing / casing for the NFCs and plant them over a myriad of tour locations, we need to ensure the integrity of our systems. No matter the circumstances of potential environmental issues, we need to make sure the NFCs can maintain a significant threshold for its functionality. Thus, we have designed a weatherproof and sustainable housing for our NFCs that will contribute to the longevity of its use case at the tour locations.

Figure 19: Left - CAD model of microcontroller housing from study. Right - Housing prototype for EMURR NFCs [23].

5.6.3.2 Mesh Networks

Many of these environmental papers discuss the impact of the ZigBee protocol over their WSNs. They often use ZigBee because of its low-power, low data rate transfers. Since it also has a close proximity, they often communicate with each other to transmit data over longer distances and to the main server, this is called a mesh network. They use this mesh network to conserve energy to transmit even more data between the nodes in the network [23].

Similar to how we will be constructing our network system in EMURR, the agricultural significance of how the networks play a role in the system will also shine light into how EMURR uses its clients and NFCs to communicate with each other. Most of these agricultural studies have shown to always connect their respective mesh networks to a larger, central server that handles the data acquisition by putting the data into a database in the cloud. Homogeneous to this central server-based approach, EMURR intends to gather data from the clients and store them in the web builder database to provide a useful tool to the tour guides to display statistics of the tour. These include NFC interactions, heat maps, and other stretch goal statistics that will help the tour guides elaborate on certain webpages that grant the most interactions and provide useful tools for improvements amongst the tours. Though the mesh networks won't be particularly useful in the EMURR unit, we can foresee the implications of how these networks interact with each other to provide useful statistics amongst their systems [24].

5.6.3.3 Importance of Wi-Fi

One particular study shows the utilitarian aspects of using Wi-Fi over Bluetooth because of its portability and strong communication characteristics. Furthermore, they make the argument of its ability to extend to not only dedicated sensors within a mesh network, but the flexibility of any sensory unit to participate in the collection of data, such as a mobile or cellular device. The versatility of how important it is to allow voluntary contribution highlights the necessity for user participation over a wireless network. In the case of EMURR, the extensibility of collaboration between its users and outside individuals who didn't intend to participate in our network is important to the connectedness of the wireless structure. In other words, users who happen to be at the tour locations at the same time as the tour guides should be able to participate and contribute if the tour guide allows. With this extra data, we allow for more random distribution when analyzing the statistics of our tour sites. In relation to heat mapping, this could be a great tool to emphasize loved ones who visit their deceased relatives, in which they can read and contribute to the existing sites. It may even enhance the experience for others to hear notes left for loved ones on the sites for the tour attendees to understand and relate to the humanistic impact of respective veterans or innovative artists. Much like how participatory sensory contributions can help the distributions in the environment, relatedness between individuals in the network can aid the experience of these tours [24].

5.6.4 Portable Raspberry Pi Scripts

As we continue talking about the related work of EMURR, it would be inexcusable to not mention the previous projects that have successfully implemented Raspberry Pi web servers. With the standard implementation of third-party repositories on popular sites like GitHub, self-projects are a huge contributor to the success of smaller forms of micro processing. Many research projects are based on the foundation of this knowledge and contributions. A couple of research projects focus on the implementation of communication between a Raspberry Pi and Arduino to manage and control smart home devices [25]. Other projects focus on the pure execution of hosting a website on a Raspberry Pi and router using port forwarding in a LINUX environment [26].

5.6.4.1 Smart Home

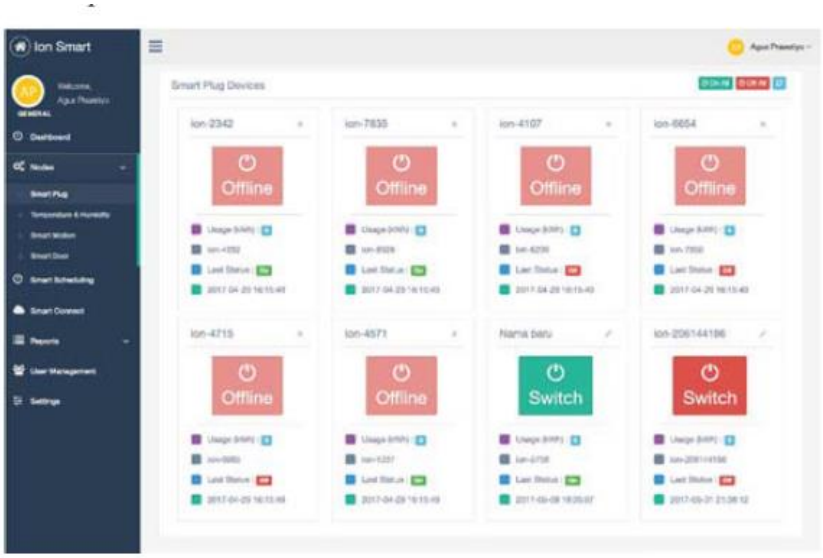


Figure 20: UI for the smart home devices using Raspberry Pi [25]

With the inclusion of many smart devices seen in the common household today, growing support for these devices blossomed throughout the smart home industry. People can use scripts hosted on Arduino microcontrollers to manage house devices such as security systems, locks, and lights. Previously mentioned, sensors can also be attached to such microcontrollers to acquire and interpret data. Furthermore, these microcontrollers can use this data to do analysis and perform actions based on security and health threats. One study uses humidity and smoke sensors to detect the air quality and outside temperatures. Without loss of generality, these Arduino microcontrollers can perform simple tasks like notifying the authorities or closing doors. The implications of these tasks can save energy, money, and a human life. While EMURR primarily deals with the interpretive information, these simple actions suggest the importance of implementing these sensors for other long-term features such as dangerous heat levels and other threatening hazards [25].

Additionally, most of these smart home devices use the Raspberry Pi as a controller to modulate, turn off, and change the states of these household smart devices to regulate temperatures and energy consumptions. In other words, Raspberry Pi hosts the states of the smart devices and displays a UI for the states of each device and the data acquisition from the microcontrollers. This idea could be useful for an admin page to manage data flow during the tour. Developing said sites under these conditions could be a useful tool to manage and document load speeds, total bytes distributed, and all users connected to the system. Furthermore, tour guides can control their tour sites more accurately and securely over their portable networks and manage the amount of traffic at the tour locations [25].

5.6.4.2 Raspberry Pi Servers

Several studies have shown the significance and proof of concept of hosting servers on Raspberry Pi's. Many sources prefer the use of LINUX based operating systems (OS) for their Raspberry Pi due to personal preferences. Though other OS can be hosted on the Pi in the case of using a smaller scaled OS like Wheezy [27].

Nonetheless, many studies have had to port forward their internet protocol (IP) address in the case of allowing outside networks to connect to the internal IP address. Generally, when dealing with normal network routing systems. People on a different network would have to connect to another network using a public IP address. The problem with allowing others to connect via a private IP address is security. With many evolving Internet-of-Things (IoT) attacks sharing private IP addresses can be a threat to a network. Without the addition of port forwarding in a system, local networks are subject to malicious script injections. Injections can scour user data and access private files sent on a network. Therefore, port forwarding is needed to deny access to any port that isn't already allowed to outside networks [28].

However, EMURR may not have to worry about this issue because all clients will be connected to the local network of the router. The goal is to default the router port to a specified IP address that all local network clients can connect to. This way, users can go to a specified IP address no matter the amount of EMURR units at a particular tour location and load the respective web page onto the clients mobile or cellular device. We can monitor all clients connected to the router through a previously mentioned admin page on the tour site to limit or blacklist unwanted users on the network.

In the case of extreme, worrisome IoT attacks at a tour location with heavy traffic, one study suggests using the Raspberry Pi as a tool to manage and prevent threatening attacks on the network. Here they mention the importance of an IP blacklist and the ability to customize the detection rates of said IoT attacks. While this is an important concept to consider, implementing more computation power on the Raspberry Pi could be problematic which could open discussion for a more hardware intensive device, and could tip the scale from hosting a server on the Raspberry Pi Zero 2 W to the Raspberry Pi 3. Such considerations might take effect during the use case process of buying and using the EMURR unit [28].

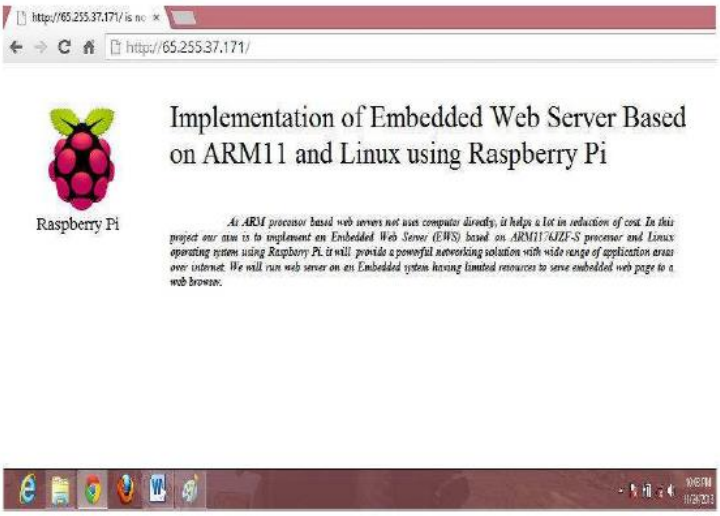


Figure 21: Server hosted on a Raspberry Pi [27].

Notably, two studies have successfully implemented a Raspberry Pi as a server on a LINUX distribution. One study used a script called RPI64Box as a portable web server for advanced RISC machines (ARM). The study discussed the importance and robust functionality of the LAMP stack and how RPI64Box and MoodleBox can open the door for more opportunities for portable server hosting applications. Some of the limitations of these applications is that they did not support more than 30 clients for their server-based systems. Much like the implied hurdles of hosting our servers to a group larger than 30 users, we may have to do more extensive research into the limitations of connected users on our router. Although, for the intended 20 users for our EMURR unit, this remains a low issue for future implementations [26].

Another study uses a true LAMP stack to host a site, but on a public network, and while this does prove the concept of hosting a webpage on a Pi, we still need our users to connect via a local IP address. Typically speaking, these IP addresses contain 192.168.X.X. Regardless of this minute issue, if we overcome the public to local IP address connection, the theory should still hold for a fluid web builder to tour site LAMP stack compilation process [27].

5.6.5 Conclusion

Though many of these studies prove the concepts of how they will implement certain aspects of these studies into the capabilities of EMURR, we must take from the conclusions of previous studies and implement and avoid the good and problematic solutions of each of the findings respectively. It has been proven that:

- A Raspberry Pi with similar or better speeds than an Android device can support the data transfer sizes within a reasonable time.
- Managing power correctly will save us effectively double the amount of power consumption from our WLAN router.
- With the right CAD models, we can create weatherproof garden stake NFC holders.
- Using a mesh network can increase the statistical effectiveness in our data.
- Implementing Wi-Fi as a base module for how we communicate over our network can open the door for other users to contribute to the much larger EMURR system.
- Including Arduinos and / or sensors as a part of the tour experience could enhance experiences or save someone.
- Hosting a tour site on a Raspberry Pi using a LAMP stack is possible.

The idea behind the analysis of these studies was to ensure the integrity of the EMURR project before building it. Overall, the results of these studies show promise in the completion and success of the project.

6 Project Block Diagrams

6.1. Tour Website Interaction Project Block Diagram

The most basic interaction consists of a user accessing various pages of the tour hosted on the Pi, which was originally downloaded from the main site containing all the tours.

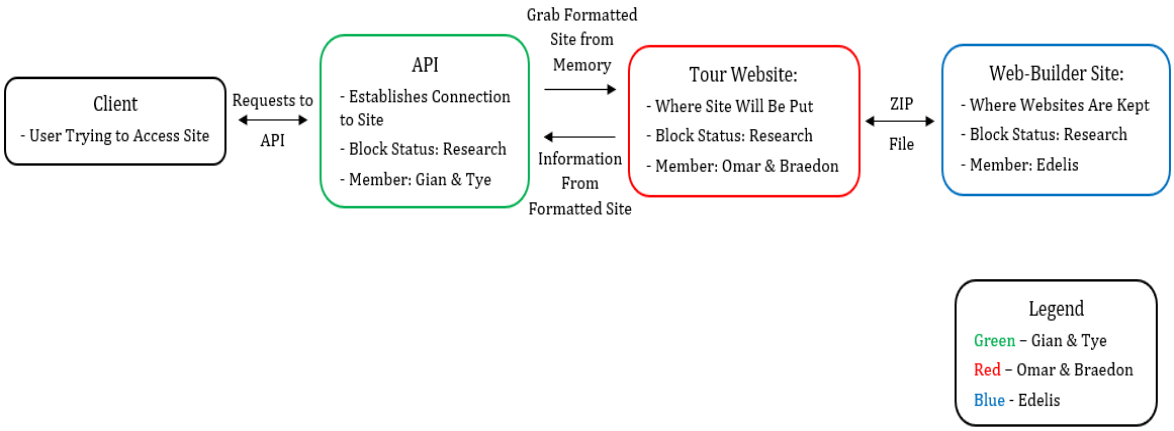


Figure 22: Tour website interaction block diagram

6.2. Web-Builder Website Project Block Diagram

Users will make API calls related to user function or processes related to creating a tour site. These calls will then communicate and get information from the database.

Figure 23: Accessing the tour site

6.3. Software Project Block Diagram

A URL that redirects to the information hosted on the Pi will be found on NFC tags. The current method for uploading a tour from our database would be through ZIP files.

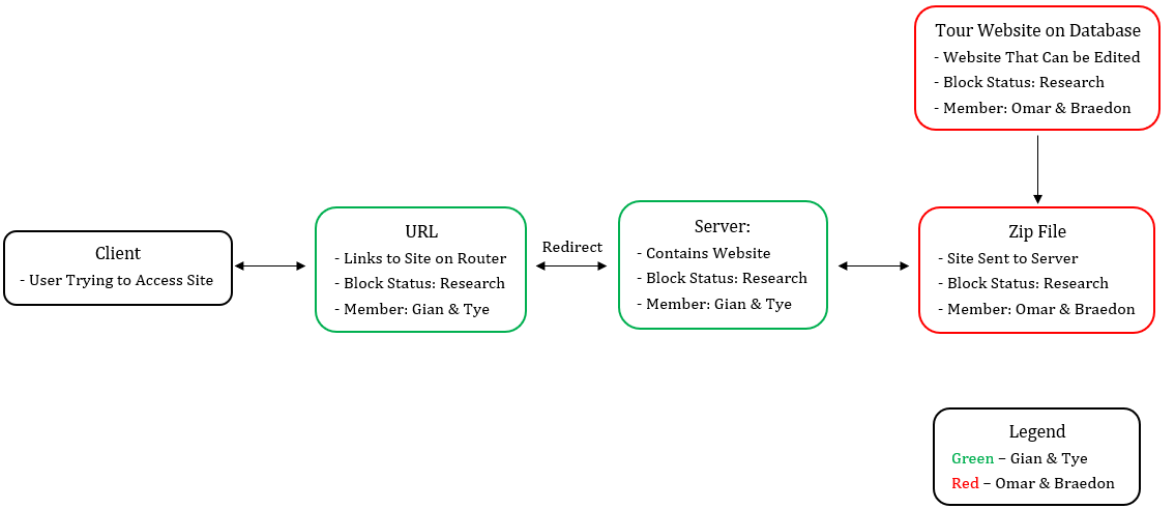


Figure 24: Accessing the site on the Pi

6.4. Hardware Project Block Diagram

Hardware communication consists of users tapping NFC tags with their phones, these tags will contain URLs that will redirect to pages on the Pi which are being transmitted by the router. The site will have to be uploaded directly to the Pi from a laptop.

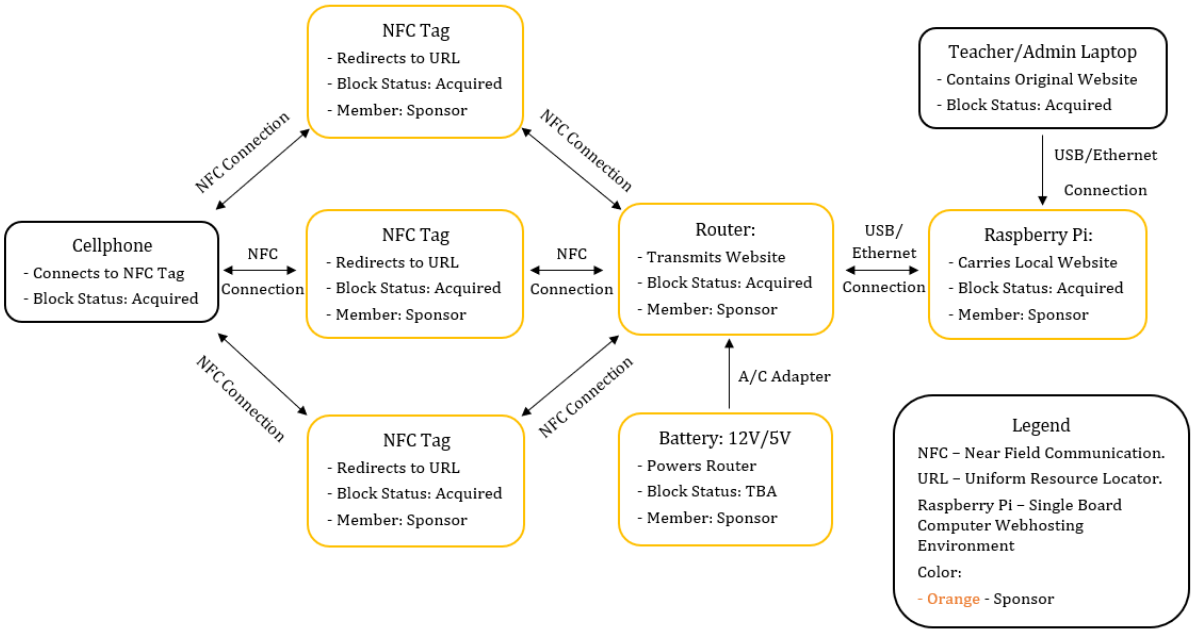


Figure 25: Hardware communication diagram

6.5. Use Case Diagram

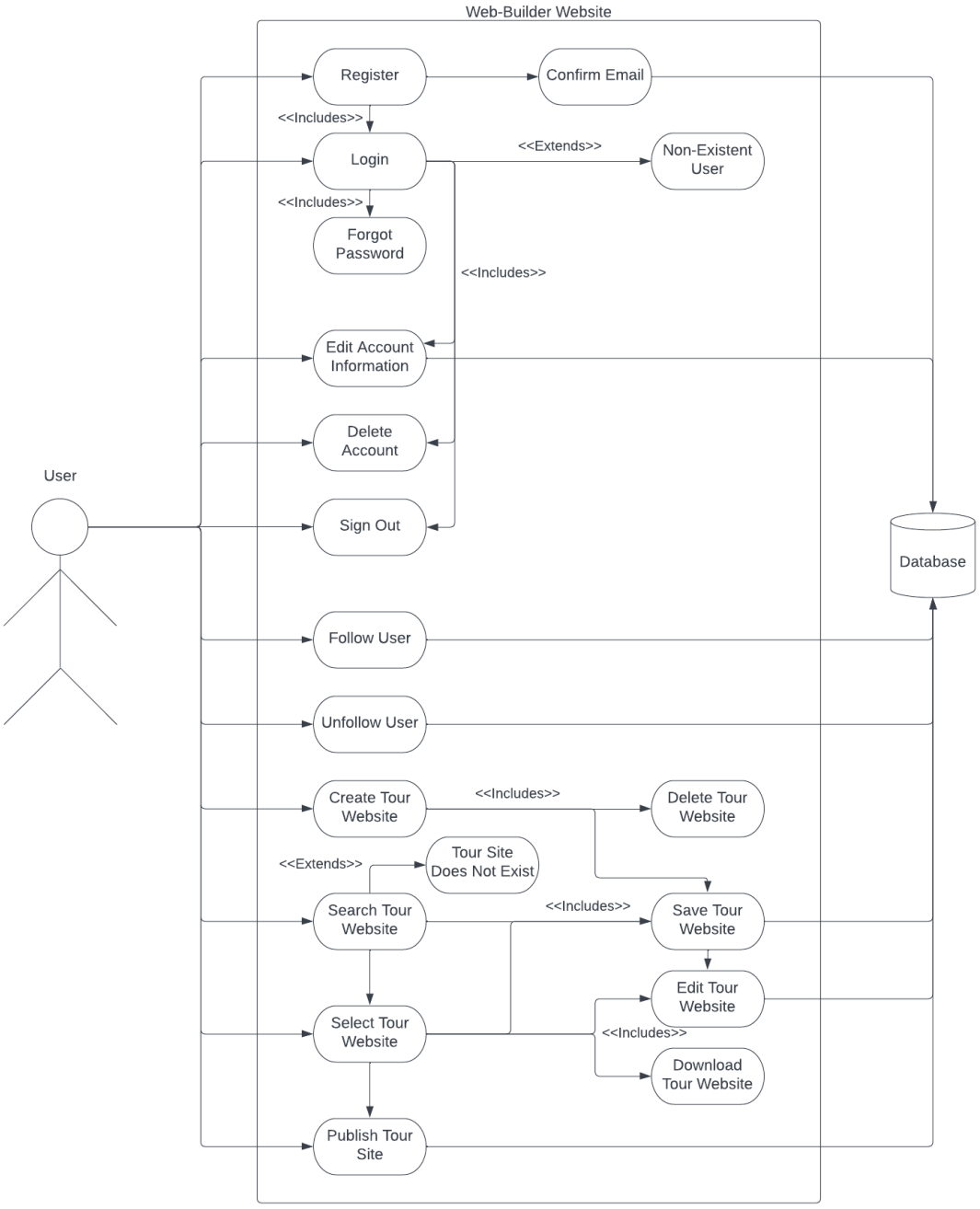


Figure 26: Use Case Diagram of EMURR application

7 Detailed Design Content

7.1. Overview

This section describes the design details of the project, and overviews many of the considerations and discussions made over the duration of several weeks. This covers the vast majority of design considerations for the project, including the API, the database, the web builder, frontend styling, testing, user accessibility, and more.

7.2. API

7.2.1 Web-Builder API Endpoints: Front-End to Database

When it comes to communication between our main web-builder website and our database it has been determined that a given number of API's need to be developed in order to ensure a proper experience. The main endpoints have been categorized into separate sections and listed below:

User Experience:

- Login.
- Register.
- Follow User.
- Delete Account.
- Forgot Password.
- Change User Information.

Web-builder to Database:

- Edit Tour Website
- Download Website
- Create Tour Website.
- Search Tour Website.
- Publish Tour Website.
- Retrieve Tour Website.
- Save Found Tour Website.
- Save Created Tour Website.

APIs related to the direct user-experience will primarily consist of basic account functions that most websites already contain. Examples such as login and register listed above, are the most essential endpoints that we will have to establish but there is room for more to be developed as the features of the app are decided upon. Because we have suggested the idea of publishing websites for other users to see, there has also been discussion as to allowing users to follow other users who regularly publish or have high quality sites.

For Web-builder to Database APIs they are primarily related to all functions around creating the tour sites, but not those that pertain to the tour sites interacting with the database. Examples include create website, retrieve a website from your database or even publish website. As we want users to make their sites publicly available, options such as search for a website and save those sites are also being developed.

7.2.2 Testing Web-Builder API Endpoints

Testing developed API endpoints for communication between the web-builder website and database will consist of two separate methods.

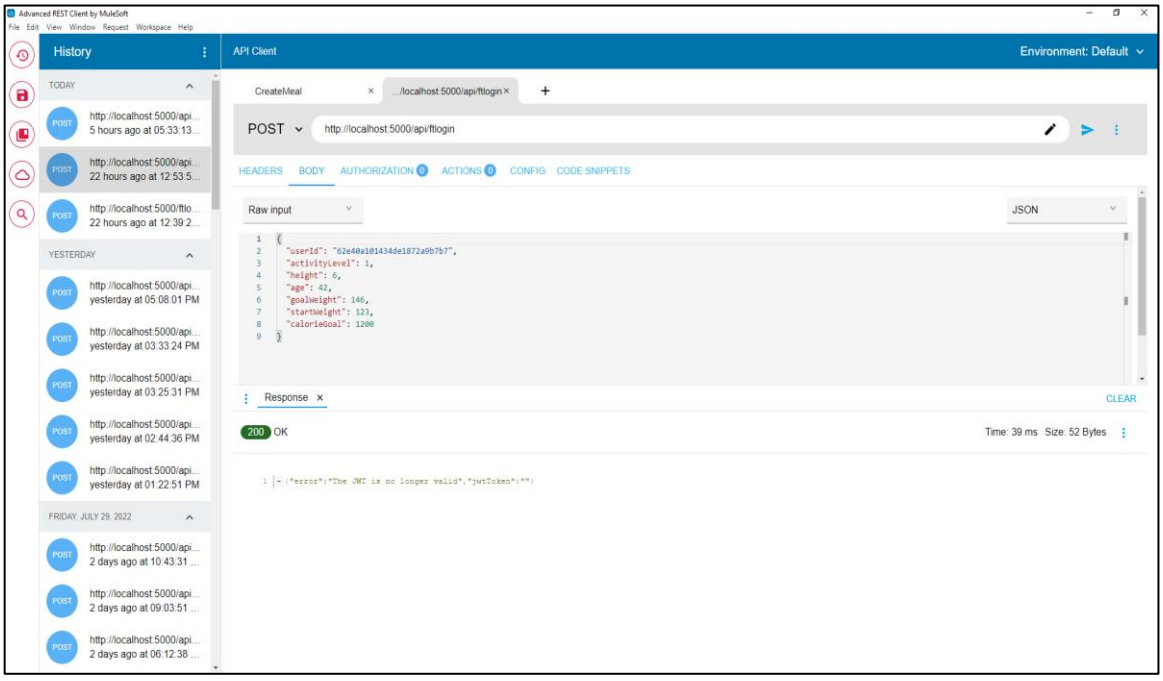


Figure 27: ARC API Testing Run on One of Our Machines.

The first method is testing endpoints through the use of programs like ARC. ARC stands for Advanced REST Client and is a free program dedicated to testing API endpoints. It provides users with an interface where they can directly connect to their sites API and send requests directly to the database. Depending on what is being sent, output can also be read directly on ARC's interface. ARC also provides the ability to customize what is being sent to the database as users can decide the HTTP method used, the header value and even the direct information being sent to the database. An example would consist of a user sending a POST request with JSON formatted user information to their register new user API. If they had their API reply with a success comment or code, this would be visible in the client itself.

To test the API using this method, those in our group responsible of API development would most likely individually check each API one after another as there is not direct access to each API in ARC. Furthermore, this testing would occur immediately after an API is developed and as a part of a final group test to ensure correct functionality. In terms of the web-builder API however, ARC would primarily only be used to test APIs in the 'User Experience' category. This is due to the fact that these API's do not require the movement or creation of files related to a made tour site.

The second method for testing endpoints is directly sending, loading, and saving tour site files on our web-builder page. Albeit a time consuming and trivial method, due to the transfer of files between our web-builder page and our database our group has decided this is our first immediate option, while alternatives are still being devised. This process will consist of making example tour sites and directly placing them in our database or filesystem. Once placed, all functionalities related to retrieving sites can be tested and monitored for correct actions. To test the creation and saving of sites, plain sites will have to be created directly on our web-builder and traced to our database after each command. As previously stated, this process is rather arduous, so more efficient methods are currently in development.

7.2.3 Documenting API Development

Albeit using ARC to test our API, we have considered many pieces of software for documenting the inputs to our API's, as well as how our endpoints generally work. The piece of software that is currently our front runner is Swaggerhub.

Used by a majority of our group members on previous projects, Swaggerhub is a platform that allows for comprehensive and in detail API documentation, as well as direct testing of API found on a given site. Swaggerhub achieves both of these goals by having a standardized language used for outlining the given attributes of an API endpoint. This language automatically formats information into concise and direct documentation that includes what each endpoint accepts as their input, the types, what functions call the endpoint and even brief descriptions of each endpoint's capabilities. The software is very thorough in its documenting abilities, which is why we are considering it, but an extra benefit is that it allows us to directly test our endpoints on the platform. Through the providing of our main URL, as well as naming each API endpoint properly, Swaggerhub creates direct connections to our website API in the same way ARC does. Due to this, depending on the ease of use, Swaggerhub will be a productive source for API documentation as well as quick testing.

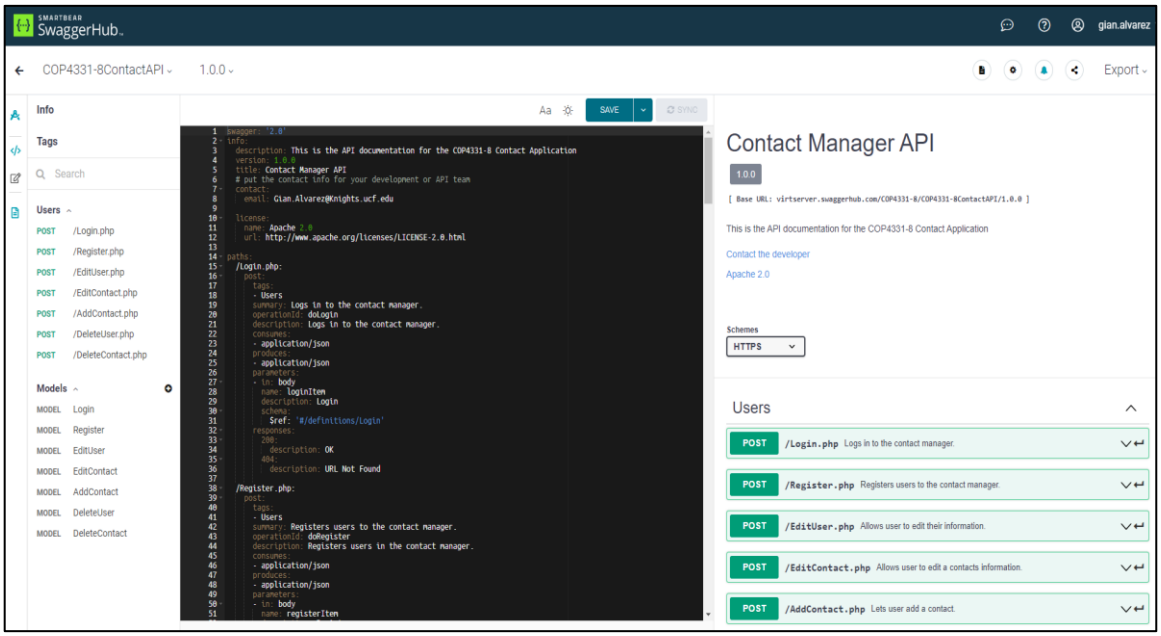


Figure 28: Example of SwaggerHub API documentation created by one of our group members.

7.3. Project deployment

As it was mentioned earlier, the team decided to use AWS's wide suite of tools to deploy the project for a number of reasons. Among those are: the free tier options of EC2 and RDS would meet the requirements of the application during development, gain real-world experience in AWS products, and easy to use services for deploying.

7.3.1 EC2 instance

The EC2 instance is the virtual server in the cloud. The free tier option of the instance is running Ubuntu Server and the type is *"t2.micro"* (1 vCPU and 1 GiB of memory) with up to 30 GB of storage. Additional details of the EC2 instance are shown in the figure below.

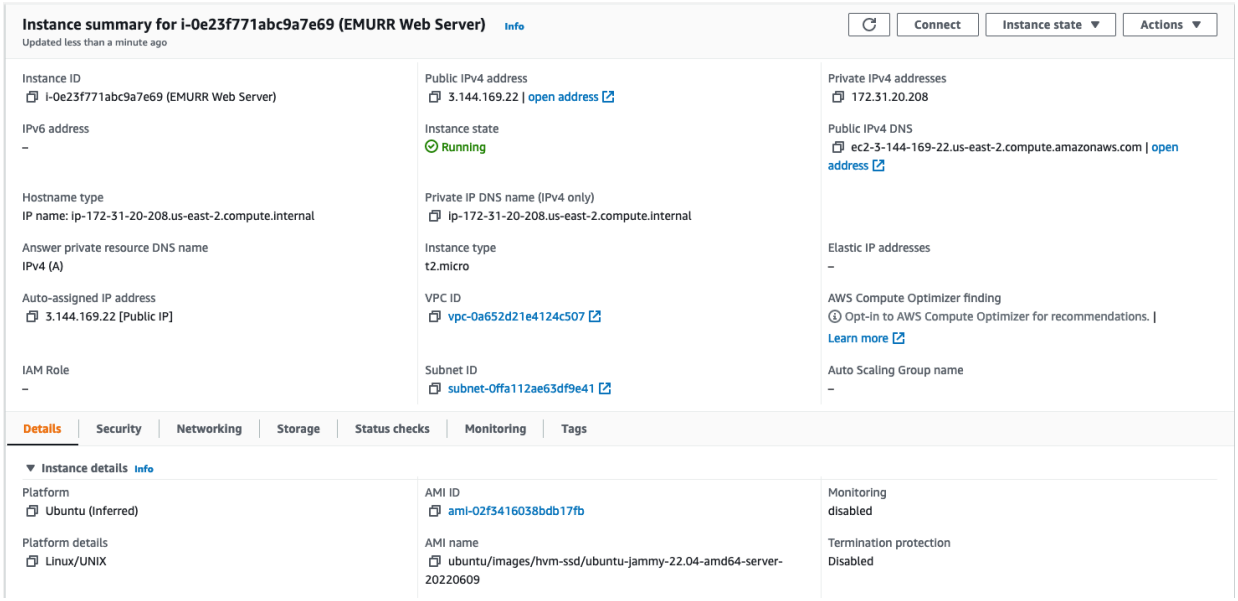


Figure 29: Ubuntu virtual server in Amazon EC2 instance

We securely SSH into the Amazon EC2 Linux instance from terminal in port 22 using a key pair. (See figure below)

Figure 30: Connection to the Linux instance from terminal

For the database, we could install and configure MySQL server in the Linux instance; however, this will require self-managing the operating system and the database at the same time and backing up the database which can be time consuming and complex. In addition, the free tier provisioned storage is limited and having a database along with the OS and any necessary files of the application will require scaling up the instance as the database size increases which could possibly exceed the free tier limits. Instead of hosting and managing the database, we can use an Amazon RDS instance to handle all those tasks for us and we can just focus on writing queries and accessing the data in the database.

7.4. Database

7.4.1 Database Schema

The Entity Relationship Diagram (ERD) below shows a visual overview of the schema of the EMURR database. It was created with the visual data modeling feature of MySQL Workbench. This is essentially a graphical representation that depicts how the data is organized in the database. The schema is considered the blueprint of the database as it lays out all the table names, fields, datatypes, and the relationships between these entities. All team members participated in this process to ensure all the required fields for our application were captured. Subsequently, a team member with the database role will recreate as closely as possible the schema on the database server. As the project continues to evolve, there will likely be modifications to the database and any change will be recorded here in the schema first.

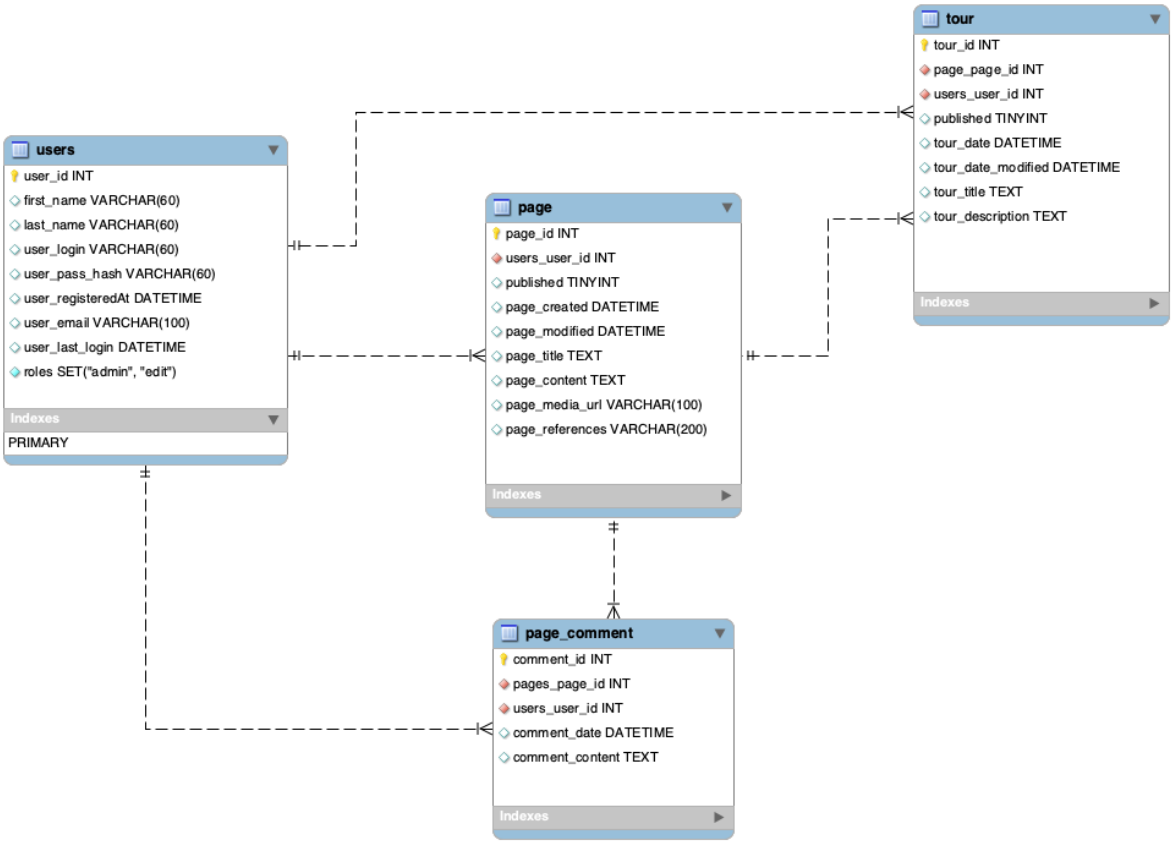


Figure 31: Database Shema (ERD)

7.4.2 Tables Overview

This section provides an overview of the tables created during the data modeling process of the project database.

Table Name	Description	Relevant Area (s) of web builder
users	Login credentials of all users of the application and their roles (Amin, Editors)	Admin Dashboard: - users > add, edit, view, delete, assign role - roles > list users - roles > set roles - tours > push to the Pi

pages	The core of the application is the pages table. Each page features information as follows: - Who created the page - Page content (title, text, media, comments, references) - Created / modified date	User Dashboard: - pages > add new - pages > Library (view, edit, delete)
tours	A collection of pages. Tours are associated with pages	User Dashboard: -tours > create new > search and add pages - tours > Library (view, edit, delete pages or entire tour)
comments	This is a stretch goal	- pages > Library > pages > comments

Table 4: Project Database Tables.

7.4.3 Database Instance

The EMURR database was set up in Amazon RDS MySQL instance -the AWS solution for relational databases- and we connect to it using MySQL Workbench. The figure below shows some basic information about the database instance in the cloud. However, when the project is delivered to the sponsors, the database will be hosted on the CHDR server.

Since we are using the free tier, there is only one type of DB instance size which is the *"db.t3.micro"*. This is a very small and lightweight instance featuring 2 vCPUs, 1GiB RAM, and up to 5 Gbps of network performance. It is not going to be able to perform work at very high throughput, but since this is just developing, we are not worried about the limitations. In terms of storage, the free tier allocates 20 GiB but enables storage autoscaling meaning that if the DB hits the upper end of the provisioned storage, it will allocate more storage resources automatically.

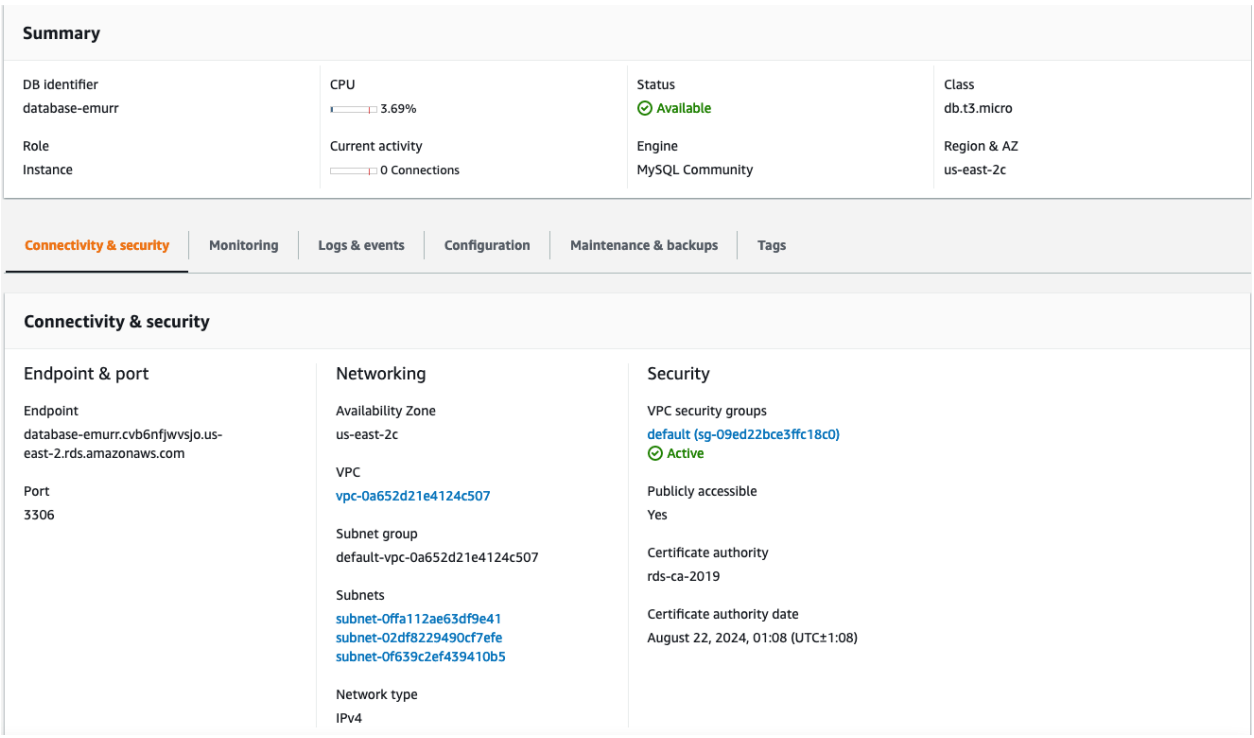


Figure 32: Database hosted in Amazon RDS MySQL instance

7.5. Web Builder

7.5.1 Introduction

One of the most crucial decisions when making a web application is choosing a framework. The framework a web application is hosted on will decide which coding patterns are used, how the pipeline handles rendering, and how we implement frontend and backend APIs. Express on the backend, handles routing in the files, usually under index{.js, .ts} which branches out into a routes folder which manages all CRUD endpoints and their respective routes. Vue on the front-end handles routes using the ‘vue-router’ which handles the routes as an object usually built in a single file. Each of these frameworks offer different tools to help developers manage and build projects. For example, React uses hooks to manage and store data in a state using getter and setter anonymous functions which can be implemented in the JSX of a React component.

On the other hand, Angular is built on TypeScript, which manages and stores data using injections which stores the data in a class and can be attached to and used by multiple components when added to the respective component's constructor. Clearly, regardless of the framework we use, the end result is the same, but the syntax and the architecture of the code changes. In this case, React manages data with hooks, and Angular with injections. Though the result for our users will ultimately stay the same, it's important to consider the different types of frameworks we will use in the project, and how we plan to implement said frameworks appropriately. Considering the full stack and the frontend-to-backend design of the application, frameworks / languages such as GraphQL will provide tools to construct proper project patterns and induce a maintainable codebase.

7.5.2 TypeScript

TypeScript is a language built on top of JavaScript to handle variable type errors at compile time. This language adds additional syntax in JavaScript, and when used with an integrated development environment (IDE) allows the user to spot syntax errors in real time. As a result, TypeScript spots errors even before compilation which indirectly results in little to no debugging. For example, when sending data through the parameters, one must specify the type as per a statically typed language. This ensures that the data is static between all function calls, and data fetches and stores.

Developers use this primarily when building web applications because it provides a clear understanding of the type of data JavaScript will use without inferring the types. One can simply trace the type of a variable to understand how to manipulate, set, and get variables throughout JavaScript without worry. Trying to dynamically type a variable in a TypeScript file while through a Type Exception in the IDE if the TypeScript plugin is installed. If the TypeScript plugin is not installed, then it will be caught when using the 'tsc' command. When deployed using the 'tsc' command, TypeScript will compile all its existing code into JavaScript. The code written in JavaScript can be used for the web application in its deployment stage. Regardless of the framework a project uses, TypeScript should be integrated in our project without question. All in all, TypeScript provides benefits to the JavaScript language for team collaboration, debugging, and readability throughout development.

7.5.3 React

React, made by Facebook, is a JavaScript library for user interface components. Mainly used for web applications, React does support mobile development which could be used as a stretch goal for our website builder. More so, React is used to manage the UI render cycle using JSX, a language designed to be written in JavaScript with syntactic sugar for writing special HTML. The HTML can be used with JavaScript expressions which will then be rendered upon React's hook or render lifecycle.

React uses a multitude of JavaScript classes to manage the render states of components throughout its lifecycle. That is until React 16.8 where they introduced hooks as a way to write JSX using React features without writing classes, using functions instead. It does this by using states to define when a component should render, extending from the embedded classes inside React whenever a mutator is called. These hooks were added to soften the workload in the developer by taking the responsibility of rendering and to create a more concrete pipeline. The mutator is a method that is called during the life cycle of a render event which is handled, in most frontend frameworks, after the asynchronous call to the backend API. Before, React components were too object-oriented in the sense that creating several children of different components could get too flustered and dependent on their parent components. As a result, developers would group these components together and manage the time in which they render to handle all cases of either component to be rendered. This is considered bad coding habits because in this case, we render two components in the document object model (DOM) when we should have only rendered one. All in all, we will be using hooks in React to manage the states of our UI. We can source and isolate the elements of our UI as separate components which will make the development of the project scalable, and produce solid, fast react elements.

7.5.4 NextJS

NextJS (Next) is a React framework run on Node that manages routing and rendering in the frontend and backend application. Because Next supports TypeScript, we will be using it within this framework. Furthermore, Next provides a multitude of plugins for managing the render states of React.

Next language similar to React hooks to manage the creation of React classes inside the project. During compile time, Next will generate the React elements for us using the static classes, minify the code to speed up processing time, bundle the code from the dependencies (node modules), and code split the bundle to be accessed in the form of different URLs. On the backend of Next, it will parse its syntax for rendering components either on server, or on the client.

Next is widely known for its use in server-side rendering. The ability to write code in both client and server-side rendering allows developers to weigh costs between processing on serverless web services and waging render times done by the client. Next provides simple tools as functions to render these components in the same file as the React hooks, which next calls props. This way, referencing the props inside the JSX doesn't have to be exported and can therefore incentivize clean code and contains the data within its relative component.

As we create the source code for the project, at some point we will need to consider the content management system (CMS). The content management system will provide tools for building the tour site on our frontend. We will then use this to save their data to our filesystem and database and load their changes to the frontend. To this point, Next offers the ability to pre-render HTML before the react components are initialized. Often used for search engine optimization, pre-rendering could be helpful for rendering components in our application related to the separation of the CMS before and after the API is called. We will, however, build and use server-side rendering to pre-render our components after the CMS API is called. Statically generating our website builder CMS would be near impossible if the user is allowed to dynamically make their sites. Though we could add static generation our default data stored in our database.

We consider using Next because it is a lightweight, powerful tool to manage the state of our React components for both the server and client. The ability to map our components and store them on the server could enhance user experience in our web application. Additionally, since the routing is done in the directory, we can intuitively name and remember the URL routes for testing and deployment. Comparatively, we must maintain a strict coding style to maintain the formality of our code. Perhaps if we all construct our pages separately in our project, it could slow down debugging, and result in the failure of the project. However, if the coding style is maintained, we should be able to organize and distribute clean components which will contribute to the growth of the project.

7.5.5 RedwoodJS

RedwoodJS (Redwood) is a full stack framework that uses React, GraphQL, Prisma, TypeScript, Jest, and Storybook to handle the development cycle of the web application. Unlike Next, Redwood uses the command line to manage and create pages, routes, APIs, and database tables. Redwood offers unique features that many frameworks do not offer. In the case of React, which is a framework, Redwood is more so an engine for building web apps.

One must use the command line to generate resources and respective files in the full stack. For example, the 'yarn rw g page home /' command makes several files in the project directory. It makes a sample JSX file using React hooks, a test file for testing with Jest, and a stories file for testing with Storybook. This page is already added to the Redwood *routes* file which is a single file route manager. In this file it has already added the respective page component, conveniently titled 'Home' to the routes file under the URL '/'. This is a lot for a single command and would`ve cost some serious development time just to make these separate files which happen to have generated all the testing files, components, and routes in the respective files for us.

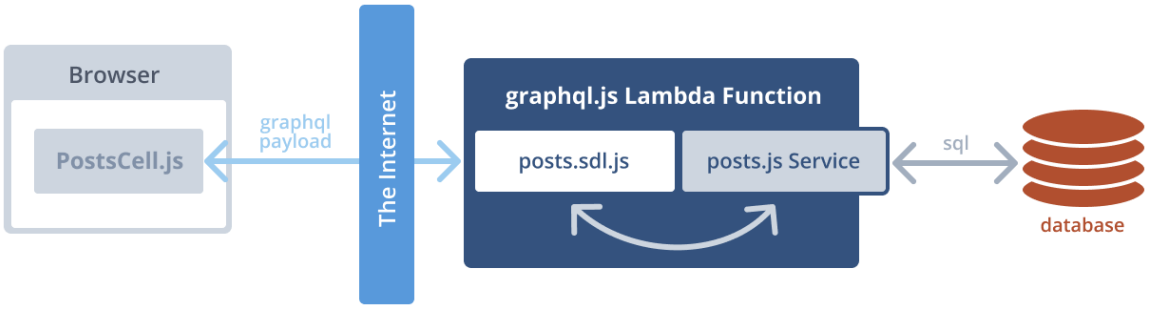


Figure 33: Data flow of RedwoodJS from browser to database through GraphQL

Because it is full stack, Redwood offers more commands. Once the corresponding Prisma file has been updated to reflect the needed changes to the database, we use the 'yarn rw Prisma migrate dev' command which creates the API using GraphQL to interact with Prisma to store data in the database. Since Prisma is used, we can use Prisma Studio to manage the database from their out-of-the-box UI. Additionally, Redwood added the corresponding test files used with Jest to manage the validity of our endpoints through GraphQL. Needless to say, the respective auth was auto generated as well.

Then, we can run a third command ‘yarn rw g scaffold post’ which will generate the corresponding frontend React components to the backend API. In fact, it creates over 10 files that all correspond to the frontend API. With this, it follows all naming conventions regarding the ‘post’ scaffold including: ‘EditPostPage’, ‘NewPostPage’, ‘PostCell’, ‘NewPost’, etc. This is already in our project. We can then create more components or restyle the page to customize this scaffold to the style of our web builder.

The previously mentioned cells that Redwood creates are components in react that load to a particular React hook when in a state of the API call. It considers four cases: loading, empty, failure, and success. Each of these states are dynamically loaded on the component when the API call is called. This way, we group the state of the component during the process of the API call respectively which should and can be considered for most web applications. EMURR would use cells to manage the states of server-side rendering during the CMS to express system errors, loading, and success cases.

There are many other features of Redwood including: specialized HTML forms as components, authentication, streamlining Storybook components, and role-based access control. More importantly, Redwood should be thought of as an engine for web development. It does most of the heavy lifting for software engineers by writing tedious, extensible code for them. However, the project is static in terms of plugin control. For a framework like Next, we can use any ORM or opt out of having Jest or Storybook for an entirely different framework. Yet, Redwood is statically based on the opinionated nature of its full stack. It would be relatively difficult or impossible to remove or add other testing frameworks or use an entirely new frontend framework like Vue or Angular. These are the tradeoffs when considering a full-stack opinionated framework such as Redwood.

7.5.6 Conclusion

Though Redwood offers great tools for modern web development, we decided to use NextJS for our full-stack framework. NextJS includes intuitive server-side rendering for front end development that can be used with our CMS for fast load times for our embedded HTML. It is generally bad practice to change the HTML in a React application because the web application is vulnerable to cross-site scripting (XSS) attacks which can be a huge problem for our clients’ building websites. Thus, we use Next to render our components embedded in our backend to prevent potential threats and ensure security within our application.

7.6. Styling

7.6.1 Introduction

Styling is especially important as we continue forward with our project. Graphic design can determine the success of a project and web application. Features such as reasonable image resolution and dots per inch (DPI), soothing color palettes, and an interactive UI will enhance the user experience for our user experience by tenfold. Users will want to interact with our site, will enjoy building their tour sites, and share their experiences with friends if they have a good user experience. Thus, we use tools to help us developers plan and implement designs to ensure the user experience is up to par with modern web applications.

7.6.2 Figma

Figma is an in-browser editing tool for designing and prototyping web pages. What makes Figma excel at web design is the suite of features which promote intuitive design and collaboration.

The design features provided by Figma empower users to make effective User Interfaces (UIs) in minutes. For example, when dragging objects on the screen displays dashed lines relative to other elements. The power of this feature is to help keep design spacing consistent relative between objects. This spacing mirrors CSS effects such as flexbox. Additionally, Figma offers an objective visual guide known as layout grids in addition to just the relative lines when dragging. Layout grids help to keep the overall spacing of the page feeling consistent and easy to reference. Figma can also simulate page transitions through on click events to help visualize page navigation. Using page transitions can help designers understand how it feels to transition from one page to another. Single Page Applications (SPAs) benefit greatly from this feature. Despite their name, splitting variations of the same page to separate frames helps to visualize transitions and transformations much more elegantly. Another feature helping designers every day is object grouping which allows for items to transform together when operated on. This is a great tool for manipulating sections of a page or Scalar Vector Graphics (SVGs) which are scalable images which can consist of many parts. When dragging or manually editing the position, size, color, or many other variables a group in Figma will all change together. Objects in particular groups, or even groups themselves, can even be locked such that when manipulations affect the page they are not affected. This helps to affect parts of a group as needed or to manipulate sections of a page together without affecting everything. The most helpful feature for web design in Figma by far is the raw CSS reference. For each element, Figma holds a property for it showing what its raw CSS would be if it were to be a webpage element. Although not always a perfect reflection of the styling necessary, it is a great way to get started or unblock oneself when styling a webpage. Merely the ability to reflect some of the CSS from an intuitive design sheet significantly closes the gap between style ideas and development ready pages.

Collaboration is another concept that comes to mind when people think about the benefits of Figma. The first and most obvious feature that supports collaboration is collaborator access. When making a design sheet Figma allows you to add other users to edit and comment on the sheet. Comments can refer to an area or object to maximize communication and minimize confusion. Like most other collaborative software, it also allows for comments to be resolved, deleted, or replied to over time.

Another collaboration feature that makes everyone's Figma experience unique is plug-ins. For any feature not given upfront Figma allows users to generate, share, and explore features made by designers for designers. This allows users to maintain a thriving ecosystem of rich features the Figma team did not think of on their own. The EMURR team is already using some plug-ins and plan on using many more. For example, one such plug-in is a colorblindness visualizer to help see page designs from the perspective of those unable to perceive colors. This plug-in provides a breakdown of the types of colorblindness with what each one will see when looking at the pages in the user's design.

7.6.3 Tailwind

Tailwind is a framework for building dynamic and customizable CSS right in the HTML. It supports frameworks such as React and Vue to interact with components and render them as raw CSS. In the tailwind config file, we will define our own tailwind features in order to support both light mode and dark mode; mobile, tablet, and desktop support.

Additionally, Tailwind offers UI snippets that can be added to the project. Some examples include sidebar layouts, tables, store pages, blog sections, headers, fly out menus, and more. We can use these UI snippets to speed up design time and ensure the functionality of our React components.

With these small but convenient features, we should create and organize our CSS to be extensible throughout the project that would otherwise be a hassle in traditional CSS. Regardless of the roles of our team members, collaborating on a portion of the project should be relatively easy to grasp the functionality of the CSS just by glancing at the components in the JSX files.

7.6.4 Conclusion

While there is no longer support for Tailwind Figma assets, we can easily simulate the conversion from Figma to Tailwind easily. Once we have planned and implemented our custom styles inside the Tailwind config file, writing the CSS should be as easy as writing the components. Additionally, we can use Storybook (more on this later) with Tailwind to validate the styling of our components separately from the page.

7.7. Testing Tools

7.7.1 Introduction

As we continue our development process, it's important to consider important tools to help us test, develop, and implement sections of the project in an efficient and timely manner. Regardless of using RedwoodJS or NextJS, we intend to use unit testing for our front end and all of our front end.

We can do so using frameworks to help us combat these issues before deploying. As a result, this ensures positive user experience and minimizes our chances for errors. Additionally, we use these frameworks to highlight other unique features during the development cycle.

7.7.2 Storybook

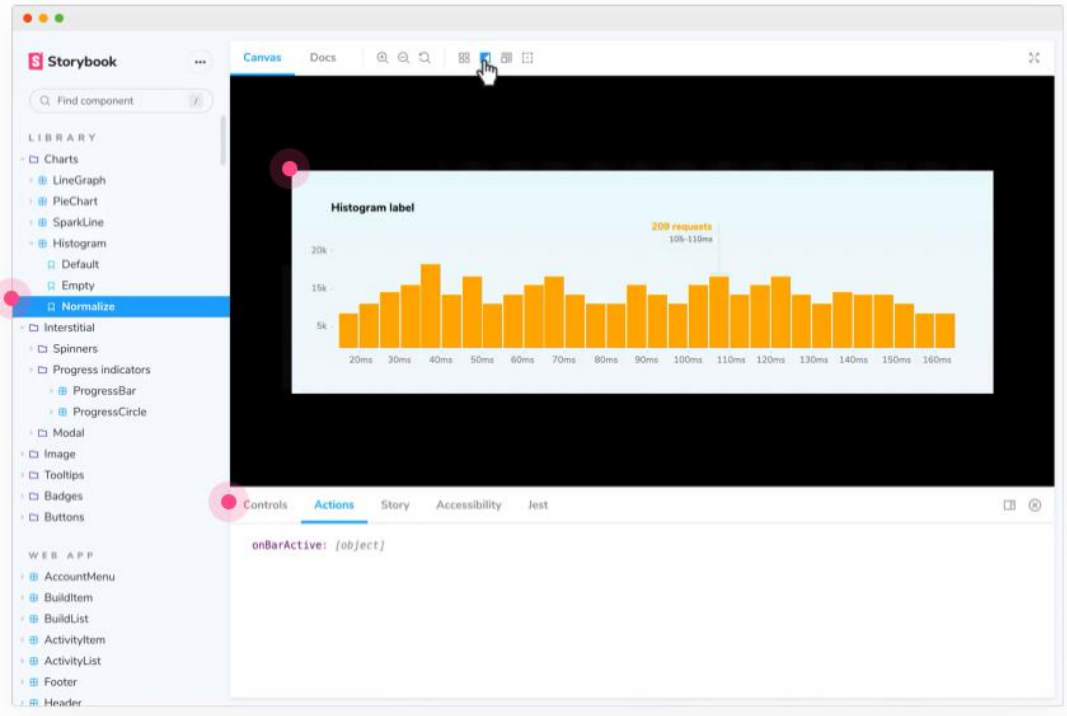


Figure 34: Storybook UI component explorer [29]

Storybook is a component and page builder for front end development. Storybook streamlines development time offers testing components in isolation, and documentation for the individual components. The purpose of this process is to create a clean, working UI used with React hooks to isolate events upon component creations. Additionally, we can investigate the button properties, adding or monitoring changes of each UI component separately. This should result in more fault tolerant web applications and encourage good coding habits. Using Storybook with a dynamic team will ensure faster code readability and improvements amongst team collaboration. Then, we can avoid fewer concerning disturbances such as creating the project in different coding styles or fixing bad architecture using copy-pasted code. In essence, Storybook will provide better team dynamics amongst the codebases.

7.7.3 Jest

Jest is a simple testing framework designed to be used for web applications. Jest will be used for our unit testing for the API calls of the website builder. We will test our Login / Registration endpoints extensively to ensure our authentication is secure and reliable. Regardless of implementing either GraphQL's or NextJS' out-of-the-box authentication library, we will ensure security for our user-base when testing with Jest using both auto and manual mocks. Furthermore, testing on the frontend will be limited using Jest in regard to the Storybook testing. Though testing will be done in both the frontend and backend, we intend using Storybook for frontend component creation and Jest for frontend to backend API request and response methods. As a result, this will lead to clean frontend components isolated for user-story driven testing and clean API interfaces by ensuring accuracy in our serialized data pipeline.

7.7.4 Conclusion

Unit testing with Storybook and Just should provide an error free web application. In practice, these tools should highlight specific components in our website builder where we can test our API calls for our user registrations and (soon to be mentioned) content management system. Testing these components will be vital to the success of our web application.

First, we will design the web builder components using Storybook to test individual functionality and the growth of the web builder components over time. Then, we will use manual tests with Jest for our web builder and how the data transitions from the user interface to the filesystem and database. Functionally, this system will produce conclusive results toward the desired goals of the project while keeping the codebase clean, reliable, secure, and consistent.

7.8. Website Builder's Building Frameworks

7.8.1 Introduction

As a result of making EMURR to coincide with our goals of architecting a website builder, we should consider using Content Management Systems (CMSs) to help us handle HTML conversions between a text editor, what-you-see-is-what-you-get (WYSIWYG) embedded editor, or other source of API to handle converting user generated websites to websites hosted on the Raspberry Pi. We consider using a CMS because it will allow users to easily interact with building websites within our web application. In some cases, we can retrieve the data they use within the CMS store in our filesystem or database. In other words, we emphasize the use of the user-story design in the frontend of our web application.

We also consider the use cases for each CMS because some are made for user-based development. For example, WordPress uses a CMS to allow their user base to build and publish their websites using their software-as-a-service (SaaS). But in our case, we need to construct a SaaS using a CMS to allow our users to build web applications in our web application that should be used and managed exclusively in our pipeline. From CMS, to storing the website in our database, to converting the storage to HTML used by the Pi, to hosting a server on the Pi, our CMS should offer us an API to interact and build for this pipeline. Nonetheless, this API should be exclusive to the process but integrated seamlessly in the system.

7.8.2 WordPress

For a sophisticated website with many options to make and customize their websites, WordPress is a modern, common tool to generate user websites with extensible plugins. Though WordPress emphasizes the ability to allow their user base to build not code with the ability to build from templates, it does propose a learning curve for most users.

Considering the case of a modern developer, we can conclude that learning a new system from the traditional LAMP stack may require a fundamental understanding of the system. Notably, we create hurdles in our system if we require a user to learn these unnatural tools. Thus, we should view some tools associated with WordPress to allow our users to access our SaaS, within our own web application using WordPress to alleviate some concerns regarding build websites in our website builder.

7.8.3 Beaver Builder

Beaver Builder focuses on adding specific templates to manage HTML components in a manageable fashion for their users. In this WYSIWYG editor there is a drag and drop selector tool where one can add, edit, customize, or delete their components on the webpage. Layouts are built to manage designs and organize the content on the web page which can be viewed and saved in real time. Once the layout is built, then one can edit the components directly on the web page. Though editing may become more complex as the project scales, they offer content area layouts to group and organize modules. They also offer header, footer, and sidebar widgets to apply to their website.

7.8.4 Divi

Much like Beaver Builder, Divi offers many of the same features with more animated and module organization options. Mentioned before, drag, and drop building will give less experienced developers the same tools for building websites without coding. Additionally, Divi offers custom CSS control and pre-built components. Consequently, users can create themes amongst their tour sites enhancing experiences and increasing interaction at the tour destinations. Furthermore, users can save, undo, and redo edits in the website builder to account for user error and ensure a better user experience upon using the platform. All in all, Devi offers many features to encourage users to build exceptional websites amongst our website builders.

7.8.5 Potential Issues

Notably, creating websites takes time already, and for a busy teacher, adding this amount of complexity to building a website is too much of a hassle, and would incentivize not using the website builder for their tour. Admittedly, this could be useful for CHDR to create more customized and dynamic websites, which teachers could then download and use for their tours. For example, we considered only allowing CHDR or an equivalent organization to manage the websites and add to the existing database to ensure website functionality during the tours while also offering a more stylized approach. The problem coincides with the centralization of this approach. For a tour guide to contribute, they would have to contact CHDR or an existing organization to add or change features on the tour site. Additionally, these organizations would have to maintain these sites which would generate recurring capital.

Using plugins like Divi and Beaver Builder also incentivize a pay-to-use method to reach better and more competent features. While this could be great for adding or enhancing the site further, we do consider the amount of complexity that would go into making a web builder that interacts with their APIs and payment methods. Thus, creates many failure points for our web application.

7.8.6 Tiny MCE

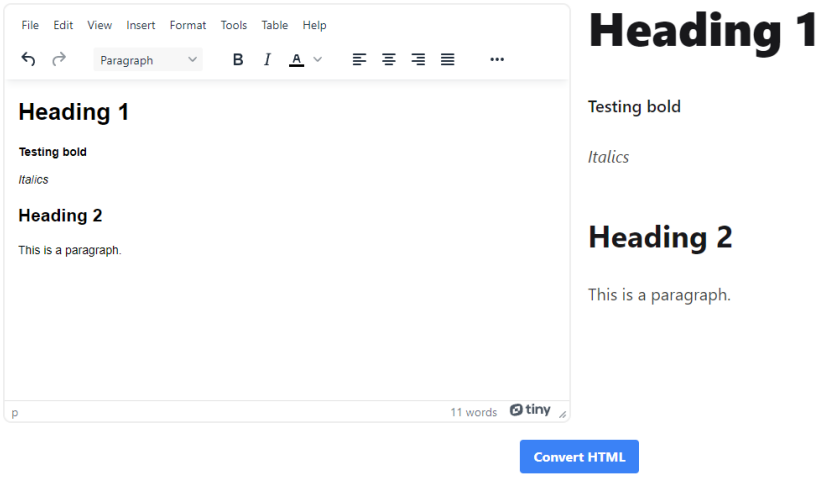


Figure 35: Tiny MCE snapshot

Tiny MCE is a CMS that standardizes a rich text editor to HTML system. They offer a multitude of ways to incorporate its API regardless of package manages, frameworks, or languages. Its design is simple, provide a intuitive and familiar rich text editor (RTE) that calls to a backend API authenticated by a given API key to convert the text generated in their editor to HTML Though there are other options like self-hosted API conversions, going through their backend should incentivize clean, workable code in the project. To illustrate, a change or update in the backend of the Tiny MCE RTE to HTML API will require minimal to no refactoring on our codebase. On the other hand, self-hosted Tiny MCE could be better for saving memory in the filesystem on the database and converting the text to HTML at website builder to Raspberry Pi compile time. Additionally, self-hosted Tiny MCE would allow us to customize certain RTE to HTML tags and modify existing ones. Nonetheless, self-hosted Tiny MCE doesn't require a payment method since it is protected under the MIT license which makes the self-hosted option more likely to be implemented in the project and has an extreme advantage to the other WordPress plugin problems. More so, Tiny MCE is intuitive in the sense that it looks much like Canvas' rich content editor (RCE) and will therefore look and feel similar to the tools teachers use to build their curriculum.

7.8.7 Conclusion

While we consider a few options for our CMS, we highly suggest using the self-hosted Tiny MCE option. This way it will incentivize teachers to use our website builder through Tiny MCE. More to the point that teachers will already be familiar with this sort of RCE, editing text in an RCE will be a breeze. Additionally, adding a few more complexities like CSS styling (colorblind mode, light mode, and dark mode) to the tour sites can be done either after the Tiny MCE RTE to HTML API call. In all cases, Tiny MCE provides many solutions for incentivizing teachers to build and use their websites on their tours.

7.9. Accessibility

7.9.1 Introduction

Accessibility on the web is an important topic to consider throughout the development of the project. Since we are making a website builder, we need to ensure that every website built is accessible to the end users (tour attendees).

Admittedly, it is incredibly frustrating to incorporate accessible features of a web application in a JavaScript framework. Therefore, we need to provide tools for not only our web application, but our sites built by our website builder. Without any interactions between adding accessible features throughout the tour site, we will generate out-of-the-box accessible features for all pages made by the website builder. For example, for those who are colorblind, there will be a color-blind mode built for the tour site during compile time. We can do this by using APIs to add such features in a timely manner. Nonetheless, every site should be accessible to all students, teachers, and tour enthusiasts.

7.9.2 Themes

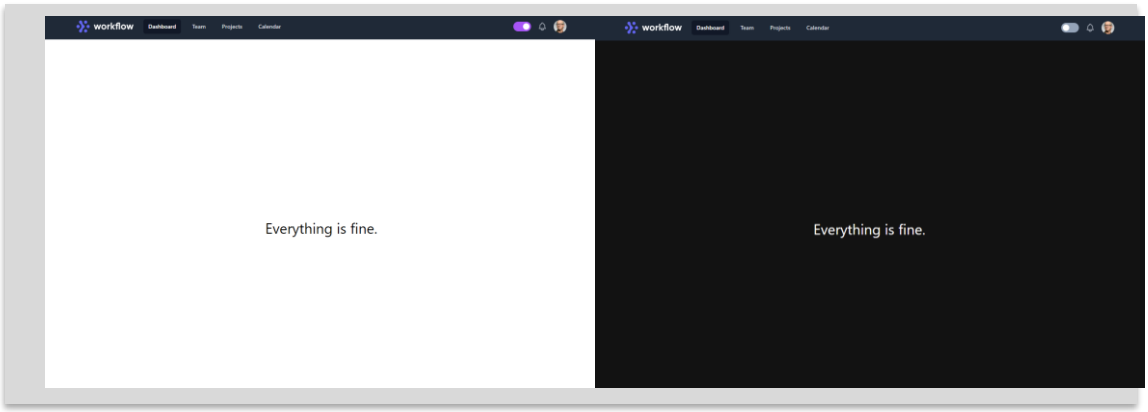


Figure 36: Prototype of light and dark mode using Tailwind

We should incorporate dark mode and light mode for the website builder and the tour site. While the tour site builder's intention for dark mode is comparatively less vital to the success of the project, it makes for an exceptional use case for most modern web users. With the increasing support for cascade style sheet themes, it is almost expected for a website of sorts to have the ability to substitute the default, more lighter color features, for a comfortable dark palette. On the other hand, we value the need for dark mode for the tour site in the case where light would be too inhibiting. For example, in the case of a museum tour, light mode may be too distracting for the more dark and ominous showcases. Therefore, causing too much distraction and eye strain for the tour attendees and people surrounding them. In the case where there is an entire classroom for kids in this proposed setting, we can easily infer the amount of light that illuminates one of these showcases would be overbearing.

Considering the dark mode on the website builder, we plan to use Tailwind's theme selector to design both the light mode and dark mode. Tailwind offers out-of-the-box design tools for using dark mode within our html styles. By using these methods, we can also borrow from their preferred-color-scheme CSS media feature, to select and default certain themes to which best fits the user. Then, in our tailwind config, we can design and pick our default color themes to coincide with our light mode color choices.

7.9.3 Color Blindness

One consideration in accessibility is color blindness. Color blindness is found in 8% of men and 0.5% of women. In other words, in a classroom of 24 students where half are male, there is likely one person to have color blindness. As we continue forward with the project this will be a concern for those on the tour and interacting with the classroom.

We encourage all of our users, including teachers to be more involved with the tour exhibits by offering inclusive color palettes. We plan to use static CSS classes during compilation of website builder to tour site process. These color themes will be statically generated and will be used over all sites built on the website builder. As a result, we limit the tools the user has to use to make a more simplified process of the user-story and development of the compilation process.

Furthermore, as a rule of thumb we intend to never convey information purely through color. A few examples of this are required text forms signified by asterisks, buttons will be signified by their shape and shadow, and links will be signified by underlines to name a few.

Another way to support accessible color use is to provide sufficient contrast between information bearing elements and backgrounds. The main EMURR page supports this in both light and dark modes by ensuring information bearing elements and backgrounds are at opposite ends of brightness with differing hues. How these hues are differentiated depends largely on the type of color blindness present in the user, all of which should be accounted for.

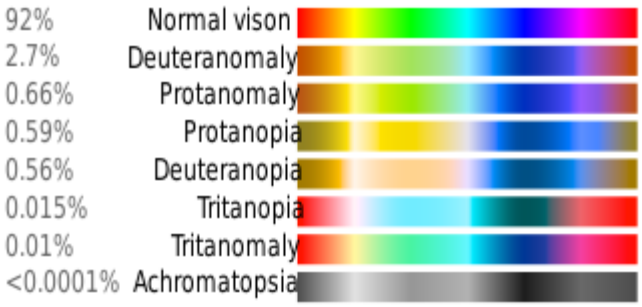


Figure 37: What colors look like to people with different kinds of color blindness [30]

The most common form of color blindness is red-green color blindness identified in figure 37 as both deuteranomaly and protanomaly. The distinction between deuteranomaly and protanomaly is mostly a subtle shift. Those with deuteranomaly have loss of green which becomes confused with red while those with protanomaly have loss of red which becomes confused with green. Naturally, confusing color combinations for this group are red-green but also brown-yellow as well as blue-purple. Protanopia and deuteranopia are like protanomaly and deuteranomaly but more severe. Those with protanomaly and deuteranomaly may have some red or green missing in their vision but with protanopia and deuteranopia the loss of distinction is absolute.

Past red-green color blindness there is also blue-yellow color blindness identified both as tritanomaly and tritanopia. Similar to protanomaly-protanopia or deuteranomaly-deuteranopia, tritanomaly is simply a less ever version of tritanopia. Confusing color combinations for these people in addition to blue-yellow are blue-green, purple-red, and yellow-pink.

The rarest case requiring the most consideration is full color blindness identified as achromatopsia. As the name and figure suggests these are people with little to no color perception of any kind. Due to achromatopsia, in addition to other forms of color blindness, it is paramount to ensure color is not the only distinguishing factor for information bearing elements.

If we were to consider the complexity of allowing users to dynamically make these color themes, we have to consider making an algorithm to make a color selector menu on the website builder; convert the color selector to a static class in CSS; and apply that static class to a range of HTML components. All of this to say, this would outrange the scope of this project. Thus, we continue forward with the three default themes for the tour site which will already be applied to all the HTML components they make and can be applied to the tour site during compilation time.

7.9.4 Subtitles

Since accessibility is a valued goal on the EMURR project, subtitles on videos will grant a tool for people who may have trouble hearing the video around them. In the case of cemeteries, subtitles are a great tool to keep students and tour attendees attentive and immersed in the touring experience. Additionally, this system promotes those who are of hearing loss where we emphasize inclusivity and helping those who need it.

Because most of our videos will be hosted in our database, loaded later on the Pi, we consider some third-party programs to convert our MP4 files or equivalent to help with the process of loading in subtitles for the videos that do not already have them. In addition, when adding a video to our database the website builder will prompt an option to either automatically add subtitles to the uploaded video or to remain as is. As a result, for the videos that already have subtitles embedded into the video or MP4, we can prevent duplicity amongst the subtitles for the videos stored on our database.

7.9.5 Slow Loads

While slow loads are important to reaching a large audience within the tour sites, we only expect them to be necessary for dynamic sites as a stretch-goal. This is because standard HTML is already optimized for fast loading times and generic loading on text but would nonetheless be a considerable asset to have within our app. Moreso, slow loading may be beneficial to have in the case of the JavaScript that will run for the CSS theme changes amongst the client-side loading at the tour sites. This way, we prioritize the interpretive information first.

Another case where slow loading may be beneficial is when loading videos. Because of the high demand of bandwidth at these tour sites with groups of twenty attendees and above, it's important to consider the load times when prompting a video. In the case of a five-minute video and a badly optimized mp4 file, it can reach a size up to 500 megabytes, which can be a lot for a Raspberry Pi and a group of 20 plus people loading at the same time. Thus, we can use slow loading to load the information first, and then, either after a delay or with a request, load the video afterwards.

7.9.6 Conclusion

With the inclusion of features like themes, slow loads, and subtitles, we provide our user base with a multitude of tools to continue using our application for enhanced tour experiences. Since we incorporate these tools with minimal overhead for the teacher or tour guide, there is no feature that is overly abstract or complicated for our tour guides to set up while building websites. Thus, we can allow our tour guides to build websites easily and comfortably without worrying about making an API, database, or any other complicated backend that they aren't already expected to know. Afterall, building these websites should be easy, and the EMURR website builder should be incorporating all of the sophisticated software engineering.

8 Planning

8.1. Project Management

8.1.1 Management Frameworks

A management framework is a meaningful consideration for any project of significant complexity, duration, or size. The team has decided to loosely follow an Agile and Scrum methodology. In order to understand the team's decision and how we arrived at it we will discuss Agile, Scrum, and as well as other considerations with our thoughts on them.

8.1.1.1 Agile

Beginning with the Agile methodology it is necessary to understand the Agile Manifesto consisting of four values and twelve principles. Ultimately the purpose of Agile is to create a flexible and sustainable environment for software development. Although it would be too cumbersome to enumerate all these values and principles Agile has become an influential part of the team and our development process. We have even taken a liking to forms of implementing Agile methodology such as Scrum and Kanban.

8.1.1.2 Scrum

Scrum is a particular framework to implement Agile consisting of five events which produce three artifacts.

Artifacts are essentially what is produced as a result of the work completed. The first of the artifacts is the product backlog. Put simply the product backlog is a laundry list of all the things that need done by the end of the project. Our team implements a product backlog in Jira for all software development topics and a backlog of documentation topics in a Google doc. The second of the artifacts is the sprint backlog. The sprint backlog is the arrangement of topics from the product backlog to particular team members for that sprint. Our team implements this artifact in Jira for software as a built-in feature as well as marking our names on documentation topics in Google docs. The third and last artifact is the product increment or result of our work. This is what we would say is a given as every week we produce results to share with sponsors and the class.

Scrum events are the steps taken to deliver the artifacts necessary. The first Scrum event is the sprint planning. At its core, sprint planning is the process of taking tasks from the product backlog, generating the sprint backlog, and determining what the goal is for the sprint. Our team implements this as a dedicated section of our team meetings. The second Scrum event is the sprint. The sprint is essentially the work done to generate the product increment. This process is a given as our team intends to deliver on sprints and working software regularly throughout the project. The third Scrum event is the daily scrum. This is essentially a status meeting, but it is meant to increase transparency and flexibility. However, our team does not find the daily scrum to fit our schedules and would become overbearing if we attempted to implement it. This event is not supported by our team. The fourth Scrum event is the sprint review. The sprint review is a meeting with stakeholders to review the product increment and update the product backlog. The team implements this through bi-weekly meetings with our sponsors on Wednesdays.

The fifth and final Scrum event is the sprint retrospective. This event is similar to the sprint review, but it consists solely of group members without stakeholders. The sprint retrospective intends to answer how the team thinks they did and how they can improve. Our team implements this after our bi-weekly sponsor meetings.

8.1.1.3 Kanban

Kanban is another practice that supports Agile project development which is centered around the Kanban board. The team intends to incorporate Kanban practices through the creatively named “Board” feature in Jira.

The Kanban board can be subdivided into five essential pieces. First, the visual signals to help identify tasks. Second, the columns each of which will represent a state that tasks can be in. Third, work in progress limits which help to maintain a consistent pace for the project. Fourth, the commitment point which identifies what tasks have been added. Fifth, the delivery point identifies what tasks have been completed and delivered to customers.

8.1.1.4 Waterfall

Waterfall is a more traditional software development practice based on strong initial design with sequential phases dependent on the preceding phase(s). Waterfall has roots in established engineering practices and saw great success in those fields. Proponents of waterfall claim it catches mistakes before development, supports greater organization, and provides higher quality documentation. However, critics claim waterfall is inflexible and wastes development time trying to predict far away problems.

Software development is still a fairly new industry and as such flexibility is a highly desirable quality. Given this fact, the team finds it essential to lean away from Waterfall methodologies if possible.

8.2. Tools

8.2.1.1 Jira

Jira is an issue tracking software released by Atlassian popularized for supporting Agile software development. There are several features the team is excited to utilize in an effort to boost productivity and uphold Agile and Scrum methodology.

One beneficial feature of using Jira is the product roadmap. The product roadmap shows tasks over time called epics. Epics are broad tasks that are meant to take one or more sprints to complete. Attached to epics are smaller items called issues. Issue types come in any variety a user may create.

Jira comes prepared with some built in issue types labelled as tasks, bugs, and user stories. Tasks are treated as simple to do items, while bugs are reconciling errors, and user stories are features described from a user's perspective. User stories implemented by the team will follow the traditional format: as a <user type>, I want to <action> so that <goal / motivation>. By virtue of the structure of senior design focusing primarily on documentation in the first semester the issues thus far have been labelled as tasks since we do not have bugs to fix nor user stories to deliver on. Within the roadmap itself, tasks and epics can be connected to show dependencies. Furthermore, a helpful addition on the left hand panel is that users can see the status of the task: "to do, in progress, done" as well as if someone in particular is assigned to the task.

As the figure below helps to demonstrate, Jira's roadmap feature acts well as a Gantt chart for product development.

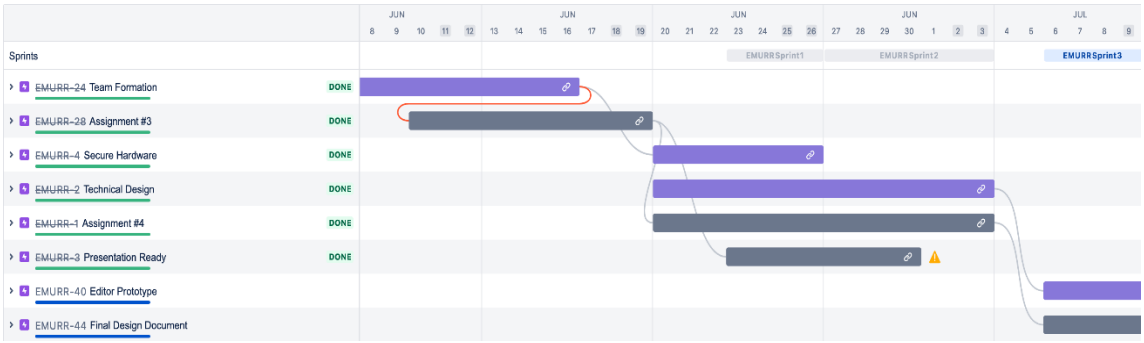


Figure 38: Jira's Roadmap Snapshot

The second feature is the backlog screen. The backlog is a list of tasks to be completed by the end of the project which can be assigned to sprints. Naturally this serves to keep separate product and sprint backlogs.

The third feature is the board screen. The board manages the status of the tasks in the current sprint between to-do, in progress, and done. While this process is reminiscent of Kanban it is helpful for managing task status.

The fourth and final feature is GitHub integration. Essentially when one formats a Git commit in a particular manner it will tie itself to a Jira issue. While this feature seems helpful it will require an unfamiliar formatting and constant awareness of the Jira roadmap by all members. For this reason, it is not a priority to implement but the team has considered it.

8.2.1.2 Discord

Discord is a Voice over IP (VoIP) application the team uses as our primary platform for communication. The main benefit of using Discord as a platform for communication is the ability to tailor servers for a particular purpose.

Servers are most accurately defined as customizable sets of text and voice channels. While users can still send direct messages to one another this server structure allows for organized public conversations between all server members around particular topics. Channels themselves are labeled and organized into categories for greater organization. For example, the EMURR server is comprised of channels such as “#announcements”, “#general”, or even role specific such as “#api”. This allows for focused conversation regarding particular topics for ease of use. Team members only have to read through conversations that they need to read through to get the information they want.

Another aspect of Discord which empowers teams for better communication is the use of pings and roles. Notifications from group chats can be overwhelming given the number of users and therefore Discord ensures notifications are given to particular user only if they want it. When someone types “@username” in their message it sends a notification only to the user at the username listed. Similarly, if a user role is created “@role” will only send notifications to users with that role. With this in mind, not only are conversations curated to be as directly applicable but so are the notifications.

Discord events are yet another feature that the team has used to boost our communication. Events are a special notification one can set up to say when and where the team will meet which has options for porting to calendars and sharing outside of the server. This has been helpful for scheduling meetings and other more irregular events.

A final feature worth mentioning is webhooks. Webhooks are essentially lightweight API’s for data sharing with Discord. The EMURR server has made use of one such webhook so far: the GitHub webhook. The GitHub webhook for Discord makes any action on a GitHub repository such as pulling, pushing, or branching into public announcements in the Discord server. In this way, the GitHub webhook is like having the contents of a well formatted `git log` continuously updated and posted to a Discord channel. This addition provides a tangible reflection of work being done to the codebase publicly available in the same space as all EMURR communications.

8.2.1.3 Version Control

Version Control Systems (VCS) are a class of software solutions designed to track changes to files and directories. Such software has become industry standard for its ability to empower developer collaboration, recover from mistakes, and overall improve workflows.

Git is the version control software the team has decided to move forward with for the project. We are aware of existing alternatives such as Perforce, Subversion, Mercurial, and many others. However, it is our understanding that the narrative around version control systems has become a statement in tech circles about Git unambiguously beating out the competition. Given this and the fact that Git is the only version control that the team is familiar with our decision was a resounding yes to using Git. By tracking changes, Git allows developers to create copies of a project in various useful forms, revert to old versions of a project, and organize code in a more human friendly manner.

Git's ability to support collaboration begins with the means in which developers can create manipulate existing codebases. First, developers can clone existing repositories essentially having their own local copy of the existing codebase. When they are ready to push their changes to the original online repository it will be ready for them to do so. This is best for a team working on one codebase together as they will all affect the same code with the same clone. Another way to create a copy of a codebase is through forks. Forks are like clones but instead of being tied to the original online repository it is a separate online repository belonging to whoever forked it. This is great for open-source projects when a user wants to take an existing codebase in a different direction than the maintainers of the project.

Lastly, another way Git enables collaboration is through branches. Branches allow developers to maintain a separate version of an existing online repository concurrently with the main codebase. Branches are kind of like the best of forking and cloning a repository at the same time. Branches allow a developer to make their own changes to an online repository while keeping the repository under its original ownership. When a developer is satisfied with the changes on their branch they can make a **pull request** which is essentially asking for approval for their changes back into the main. If the original repository owner is satisfied with the changes then the pull request is incorporated and pulled into the main codebase. This does present the opportunity for **merge conflicts** to crop up. Merge conflicts are when multiple changes have been made to the same code concurrently. The person approving the pull request and performing the merge must sort through which changes to keep and which changes to throw away. Although this process can be seen as tedious and troublesome it is a small price to pay to enable large teams to work on a singular codebase.

Forking is the process by which a developer can have their own snapshot of an existing codebase that they can contribute to on their own. Branching is also a snapshot of an existing codebase but once pushed to an online repository it will merge back into it. Although Git attempts to resolve differences on its own, often developers will run into something known as merge conflicts when merging. Merge conflicts are the differences that cropped up in lines of code while they were changed concurrently. Merge conflicts not resolved automatically by the software must be manually fixed by the developer. Although this may be difficult at first, branching allows for complex workflows and supports many developers working on the same source code.

8.2.1.4 Git Hosting Services

In order to be meaningfully beneficial, people often want Git repositories to be accessible from the web using a Git hosting service. In this way, collaborators can share and push their changes remotely. There are several services considered for doing this we have explored: GitHub, GitLab, and Bitbucket.

GitHub is well known as the popular, nigh default Git hosting service. GitHub's main advantage is ease-of-use and pervasiveness. GitHub is mostly a simple tool that takes very little time to learn once a user has acquainted themselves with Git. With that in mind, GitHub actually remains quite flexible for advanced users with additional tooling and configuration that which is not default. Therefore, allowing beginners to have a small learning curve while allowing opinionated experts to customize their experience.

Additionally, GitHub has some of the best performance among Git hosting services due to their powerful internal tools. This makes it easy to get instant feedback on result status of operations such as pushing to a remote or hosting a GitHub Page. It is thanks to the smooth performance and pervasiveness that it is the only Git hosting service the entire EMURR team is already comfortable with. In fact, GitHub has seen a lot of popularity not just with our group but in the industry. Because of this popularity GitHub has seen the largest amount of support, in comparison to its competitors, in the form of tutorials, documentation, and online help Should the team run into any issues using GitHub we will undoubtedly be in good hands. Furthermore, due to the prevalence of GitHub, hosting the project with would only help lay groundwork for the future prospects of all group members.

GitLab is a GitHub alternative known for its capacity to support Continuous Integration (CI) and Continuous Development (CD). Continuous Integration refers to continuously updating and maintaining a central codebase while Continuous Development refers to ensuring the application is constantly in a functional, development ready state. GitLab supports CI/CD automatically out of the box by constructing a pipeline to build, test, and deploy from the repository.

While building and deploying the codebase are fairly straightforward there is a bit more to talk about with testing. The automated testing includes tests for security compliance, performance, code quality, and more. All of this happens as a part just before completing a merge so developers can approve whichever changes they find salient. In order to take advantage of these benefits all one has to do is to follow documentation to set up the `.gitlab-ci.yml` file.

Default GitLab also helps with project management in numerous ways. One such way is that GitLab implements a Kanban board out of the box. Additionally, it also provides an issues management window, similar to the one used in GitHub. Furthermore, code changes can be assigned to collaborators and even milestones in the development cycle. In this way, GitLab positions itself to be an all-in-one solution for Git hosting by taking the initiative to solve tangential problems that come with hosting like CI/CD and project management. As a result, GitLab has been rising in popularity and presents itself a fierce competitor to GitHub.

Bitbucket is another option for Git hosting known for its unlimited repos and integrations with popular project management tooling. Where GitLab and GitHub both provide paid models for a number of private repos Bitbucket refuses such a model. Instead, Bitbucket's only constraint is team size. As long as there are a maximum of five collaborators on the projects Bitbucket allows for an unlimited number of private repositories. This is great for small teams working on proprietary software in the early stages. Although software licensing does exist it doesn't provide any peace of mind knowing the details of one's software solution must be public without a fee. Furthermore, Bitbucket was acquired by Atlassian in 2010. The significance of this being is Atlassian also owns popular project management software such as Trello and Jira. That is to say Bitbucket integrations with these project management solutions are head and shoulders above other Git hosting services. This can boost team productivity as they spend less time wrangling project management software and more time on development itself.

Bitbucket is also a simple solution for Git hosting without much clutter or extraneous features. Some have criticized it for being too simplistic and featureless but over time it appears its popularity and feature list has grown alongside its project management integrations. Considering the team's use of Jira and the simplistic nature of Bitbucket it will be a worthy consideration for hosting our repository.

Ultimately the EMURR team has made the decision to move forward with GitHub. Although other options presented some tempting features such as CI/CD or project management integrations we felt GitHub was the perfect mix of familiarity, simplicity, and customizability. Between ourselves and our sponsors we know the layout, interactions, and the workings of GitHub fairly well and this helps to let us focus on more development in the future. GitHub also doesn't get in the way of development with great performance and straightforward setup. And of course, any features we would like to see in the future we are certain we can find a way to integrate them into GitHub as needed. Finally, even if GitHub is not the answer, we are looking for we are certain that a switch to some other Git hosting service will be of no great effort. It will simply be a matter of changing the remote and getting accustomed to the new environment.

8.3. Workflows

Git and GitHub as tools like any other can be used in whatever manner the users wish and as it turns out there are many different ways to use them each with their own pros and cons. The team has mainly explored two separate styles of using Git which are often referred to as workflows.

The first workflow explored is git-flow. Git-flow consists of a master/main branch, a release branch, a develop branch, and separate feature branches for each feature currently being worked on. The particular variant of git-flow we explored was one that implemented a hotfixes branch. Any singular feature branch is meant to hold a singular feature on it from its assignment in the sprint backlog to its completion. The develop branch is meant to hold collections of features until it is ready for a major release. The release branch is to hold versions of release ready software until it has been tested and bugs have been fixed. Once the release branch is clean it is merged into master/main to be shown to shareholders i.e. project sponsors. Although, accidents happen and so a hotfix branch is introduced any time master/main branch code is found to have a major point of failure which is worked on until resolved.

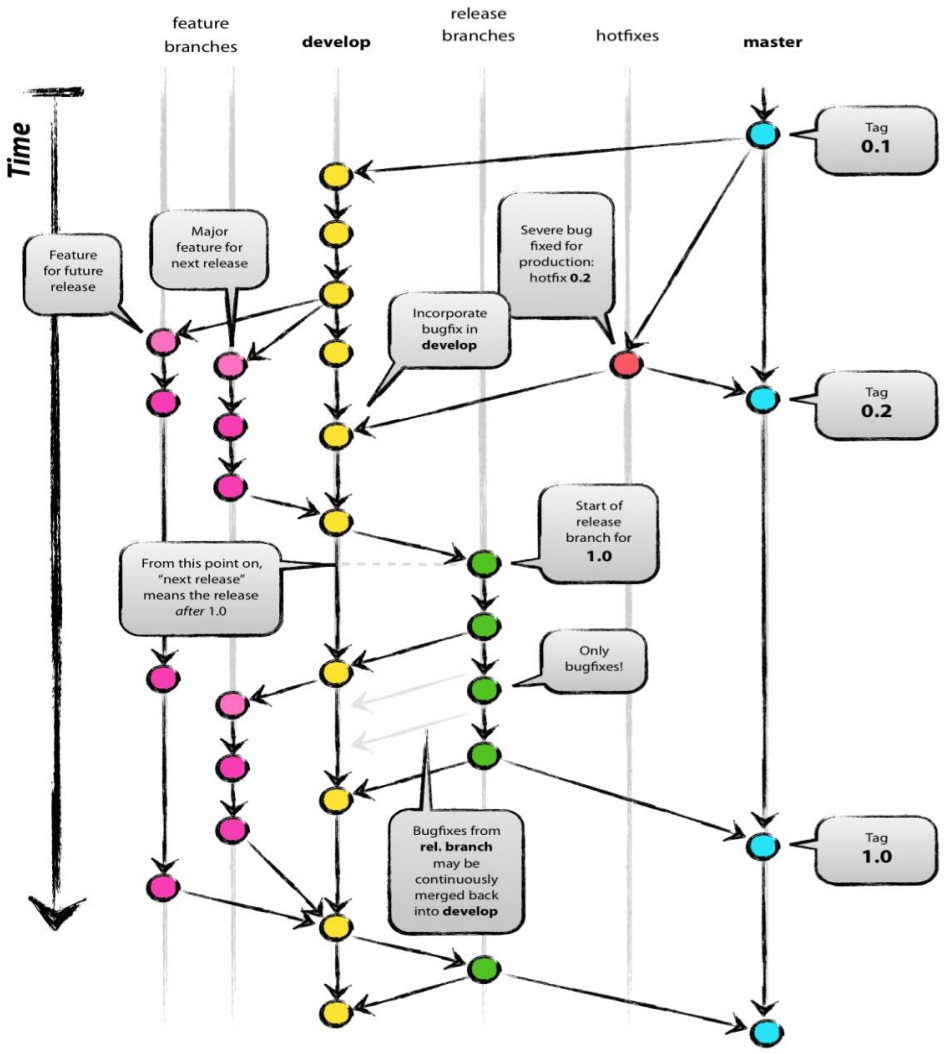


Figure 39: Gitflow Workflow [31]

The intended effect of using gitflow is that the developers' contributions are modular and distanced from the main branch due to the layers between. This is a positive aspect when there are many collaborators that are not well trusted. For example, open-source projects or companies hiring many junior developers will feel the benefits of this workflow the most. Furthermore, projects with large existing codebases will be understandably protective of their established work and prefer this workflow for that reason. However, this is not as preferable in scenarios where developers want to touch the main codebase in its entirety and do so often. When working with many experienced developers or when starting from scratch then it is preferable to often make dramatic changes to the main codebase.

To contrast, another workflow explored is trunk-based development. Trunk-based development consists of master/main, features, and releases. Features are pulled out from main and merged back in when finished. When a potential release is ready at the end of a sprint or other arbitrary goal it is branched into a release branch. Release branches hold release ready versions and iterated upon for bug fixes and minor changes for as long as they are supported.

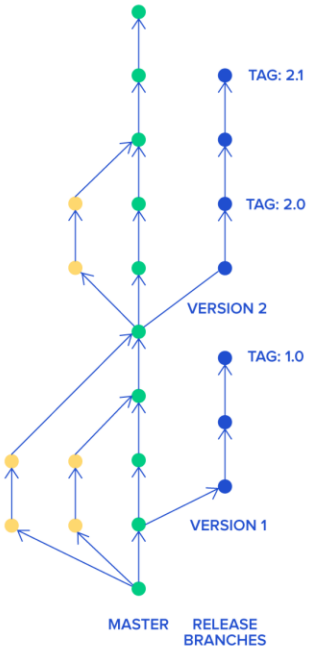


Figure 40: Trunk-Based Workflow [32]

The intended effect of using trunk-based development is to allow for rapid and large-scale development across the entire codebase. This is a positive aspect if the project is starting from scratch and one trusts those developing the project. For example, a startup consisting of many senior developers will feel the benefits of this workflow the most. Additionally, this workflow supports a faster pace of development since there are fewer layers and pull requests to pass through in order to have an effect on the release product.

Ultimately our group decided on git-flow because of the added safety of layers between features and main. The team is generally new to large scale software development projects and a safety net will help more than it can hurt. Furthermore, the modularity of commits will match the style of development seen in technology such as React. Despite concerns of not being able to develop at the same pace of trunk-based development the team believes that safety of the production codebase is more important.

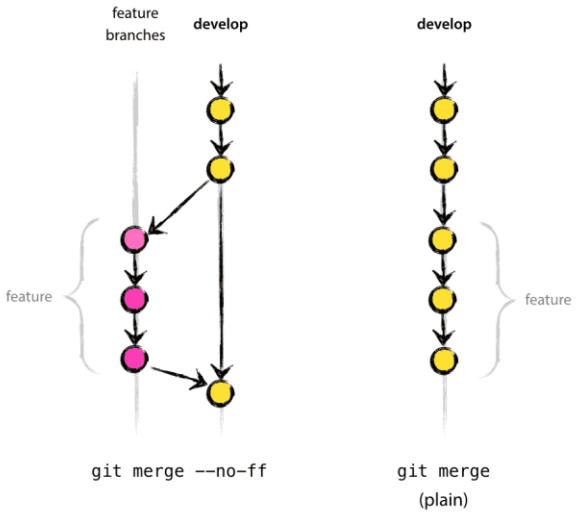


Figure 41: Git Fastforward [32]

The purpose of this flag is to preserve the branching structure of the repository even after it has merged. In Git, when there have been no changes between the branch beginning and the branch merging, it will “fast-forward”. Essentially fast-forwarding flattens the branches and shows as if they had never existed in the history of the repository. If however the team is explicit about disabling fast-forwarding then this branching history will be preserved, making it easier to reference and visualize. Therefore, the team is encouraged to set a Git alias for `git merge` to `git merge --no-ff`. A Git alias is simply a way to rename commands such that when one command is run another is executed instead. This allows for a more personalized experience, such as the team’s preference to preserve branching history.

8.4. Meetings

Meetings and how they are implemented can be a fundamental tool to the future of a team. Early in the project, our team established the significance of one weekly meeting on Sundays as a group. Similarly, our sponsors wanted to speak with our team at least bi-weekly on Wednesdays.

The weekly Sunday meetings follow a co-working format from 12PM to 2PM. A co-working format for these two hours means that during this time the main priority is working on documentation or development, as opposed to any particular subject of discussion. The purpose of this is to prioritize development while producing an environment with lower barriers to starting conversation about relevant topics. An additional side effect of this meeting is to have better optics on how far members are along in their work with natural opportunities to offer and ask for help if someone is behind. Often in Sunday meetings the first hour naturally devolves into necessary discussion. However, after the first hour the meetings fall into a focused cadence with each group member hard at work doing their necessary tasks. The original design of these meetings ended with a Scrum sprint retrospective and sprint planning, but this has since moved to Wednesdays.

The bi-weekly Wednesday sponsor meetings are essentially a sprint review from 1PM to 2PM. Like all sprint reviews we meet with our sponsors / shareholders and show what we worked on. They give us feedback, edits, and update necessary tasks as in a product backlog. Once this meeting is finished our team does a sprint retrospective talking about development process and any changes we would like to make moving forward. After which we close out with sprint planning as in pulling tasks from the product backlog to our sprint backlog. At which point the next sprint has begun and we are off to development.

8.5. Prototyping

8.5.1 Portable Tour Pack

The most essential feature of this project is the ability to connect to an offline, interactable website made available through a router while internet access is unavailable. Thus, our prototyping will begin with this feature. The minimum requirement for this prototype to be considered successful is to use a mobile device to connect to a static HTML page hosted by a Raspberry Pi through the internet router using a URL found by scanning an NFC tag.

We will also test NFC features, such as reading, writing, and locking. The team has one router, one NFC tag, and one Raspberry Pi, so this initial prototyping will be a team effort.

8.5.2 Website

The second most essential feature is the ability to create a web page and to export it to the Raspberry Pi. Our second prototype must contain a user-friendly feature to export a tour site onto a Raspberry Pi.

The last crucial feature is the web page editor. While we are not reliant on device access, this will also be a team effort. The front-end developers will create the user interface and send requests through the API, while the backend will act on the requests and use the database to store, load, and export user content.

To bring it all together, the team will create a final prototype containing all the essential features. For this prototype to be considered successful, we must be able to create a web page using the WYSIWYG editor, export it to the Raspberry Pi, broadcast the page using the Internet router, connect a phone to the router, and scan an NFC tag to access and view the page.

8.6. Testing

Testing is a crucial part of the software development cycle. Software bugs can range from mildly amusing, such as rendering issues in very specific circumstances, to extremely critical such as data corruption or security breaches. Thus, every good development team must practice robust testing.

To maximize efficiency in testing and subsequent debugging, testing frequently is recommended. Crude testing during development allows more obvious bugs to be detected and fixed quickly, without wasting time later trying to track down the source of bugs in the development cycle. Crude testing is not proper testing, and more robust testing, such as unit testing, should be conducted to ensure the software behaves as expected. We plan to do robust testing at least a few days in advance of sponsor meetings. This will give us time to fix any bugs that may appear and ensures we can provide a clean and professional demonstration to our sponsors without the disruption or embarrassment of bugs.

There are several types of testing, mostly falling into two groups: functional and non-functional testing. Functional testing is related to how the program functions, testing the correctness of software behavior. Functional testing includes unit testing, integration testing, system testing, and acceptance testing.

Non-functional testing is related to other aspects of the software, such as performance, usability, and scalability. Non-functional testing includes security testing, performance testing, usability testing, and compatibility testing. There are other types of testing such as ad-hoc testing, black box testing, and example testing.

When problems are found and cannot be resolved straight away, the team will use Jira to track issues. The problem can be prioritized according to its severity and can be discussed or resolved by any team member.

8.7. Evaluation

The project will be evaluated on a weekly basis during team meetings. Our sponsors will also evaluate our progress every two weeks during sponsor meetings. During these meetings any participant may ask questions or make suggestions to fix or improve the software.

The team or any member may also evaluate the project at any time, by reviewing and testing feature branches in our GitHub repository, and by communication through Discord. The project may also be evaluated before deployment to the development branch and the primary testing server.

9 Sponsors, Consultants & Technical Support

9.1. Sponsors

9.1.1 Amy Giroux. Lead Sponsor

Amy Larner Giroux, PhD, is Associate Director of the Center for Humanities and Digital Research at the University of Central Florida (UCF) and a Digital Historian for the National Cemetery Administration, part of the Department of Veterans Affairs. Her research involves the contact zone between humans and technology within the intersections of history and learning. By leveraging technologies such as Augmented Reality and Virtual Reality, she brings learning to both classroom and field. Her specialty is cemetery research, and she is involved in both recording the physical aspects of the space through technology such as drone photogrammetry and 360^0 imagery, and also researching the lives of those interred through genealogical research as she is a board-certified genealogist. Her research in St. Augustine National Cemetery has brought to light new information on Civil War soldiers, Native American prisoners of war, and Seminole War history. Giroux earned her doctorate in Texts and Technology from UCF and her dissertation included a digital project which mapped historical artifacts geo-spatially on Google Earth. She is currently the technical lead for UCF's Veterans Legacy Program [33].

Dr. Giroux is an experienced Senior Design sponsor with an excellent track record. She has previously sponsored twenty-five Senior Design projects, all of which were successful. She has been routinely praised as a sponsor by Senior Design instructors.

As the lead sponsor, Dr. Giroux has specified and clarified requirements for the project. She has been present in almost all of our sponsor meetings, held once every two weeks on Wednesdays. Her guidance has been instrumental to refining the details for our design project, and she has referred us to other members of UCF CHDR for technical support and consultation.

Dr. Giroux has also provided the team with physical resources, such as the Raspberry Pi and the internet router.

9.1.2 Evan Wallace

Evan is currently a 3rd year student in the Texts & Technology PhD program here at UCF. He holds a Master's degree in History from Appalachian State (2018) where he worked in the Digital Scholarship & Initiatives department, prior to that he got his Bachelor's degree in History and Religious Studies from UCF in 2016. His technical background is in front-end web development, digital media creation (Adobe Creative Cloud), and web-based analytics. In CHDR, Evan has been involved in the PRINT project based out of the department of history here at UCF. In addition, Evan has contributed various graphics, web content updates, and other maintenance related tasks to various projects in CHDR [34].

Evan Wallace is credited as the person behind the idea of EMURR – in Dr. Giroux's words, EMURR is his brainchild. Evan has been in frequent communication with the team and, similarly to Dr. Giroux, has attended almost all our sponsor meetings. Evan has answered the team's questions to the best of his ability while Dr. Giroux was unavailable during the first few weeks of the design process.

9.2. Technical Support

While Dr. Giroux and Evan Wallace are both technically minded individuals, the design team also has access to some of Dr. Giroux's colleagues at CHDR for technical support and consultation.

9.2.1 Connie Harper

Connie Harper is a software engineer working at CHDR. Connie acts as a server administrator for the CHDR server, and she is one of two technical consultants available to us through email and our Discord server. She created user accounts on the Linux server for the team members for SSH access and a user account on the CHDR MySQL server we will be using. She also provided guidance on securing our accounts and setting up public web pages on CHDR.

9.2.2 Brook Miller

Brook Miller is an applications programmer working at CHDR. He is the second technical consultant available to the team and serves an advisory role to the team.

Brook is also a sponsor for a Computer Science Senior Design project named the "Booker Prize Project" and oversees a fellow team of our peers.

9.3. Other consultants

During our communication with our sponsors, the issue of copyright laws came up. As neither the team nor our sponsors had a comprehensive understanding of copyright, we were referred to Sarah Norris from the UCF Library, with whom we scheduled a meeting to ask questions and discuss our concerns.

9.3.1 Sarah Norris

Sarah Norris is a Scholarly Communication Librarian working at the John C. Hitt Library at UCF. She leads the library's scholarly communication and open access outreach efforts [35].

Sarah has earned a Master of Library and Information Science degree from Wayne State University and her Bachelor of Science in Public Relations from Northern Michigan University. She previously worked as the Bibliographic, Access & Metadata Services Librarian at New College of Florida. Prior to that, she's also worked as a Library/Media Assistant at Glen Oaks Community College and the Three Rivers Public Library.

10 Budget and Financing

EMURR currently has no allocated budget for the software component of it as the necessary resources are open source and available for free. This will be re-evaluated as necessary in the future if the sponsors opt to use an external cloud service provider to host the website and the database, but the intent is to host both on the CHDR server. For the hardware infrastructure, the team will strive to keep the cost low and provide a cost-effective solution for underfunded organizations and community groups.

The hardware components for the project are shown in the table below. The manufacturer's suggested retail price of the microcontrollers selected for testing is $15.00 for a Raspberry Pi Zero 2 W and $45.00 for a Raspberry Pi 3 Model B+. However, the current global chip shortage has significantly impacted the price of microcontrollers and this issue has been factored in the estimated hardware cost of the project. As such, we have prepared the budget for the project taken into consideration current market prices of Raspberry Pis being sold on sites like Amazon and eBay.

Hardware Components	Pi Zero 2 W	Pi 3 Model B+
Raspberry Pi	$120.00	$230.00
Router TP-Link Archer C1200.	$100.00	
Battery to power the setup	$70.00	
MicroSD Card for Raspberry Pi 128GB	$30.00	
Micro USB Cable to power Raspberry Pi from router	$4.00	
Male to male 5.5*2.1mm DC barrel jack power cord	$8.00	
NFC Tags NTAG 215 (50 Stickers)	$15.00	
Optional: MicroSD Card reader for computer	$8.00	
Optional: 3D Printed Accessories for backpack mounting	$10.00	
Optional: Raspberry Pi Case	$10.00	
Optional: backpack for docent/exhibit guide/chaperone to house battery and router	$40.00	
TOTAL	**$415.00**	**$525.00**

Table 5: EMURR Bill of Materials (BOM)

11 Facilities Used

Throughout this semester, our group has used a short list of facilities during the creation of our project. A brief description of each location, as well as what they were used for can be found below.

11.1. Senior Design Lab

The primary facility our group regularly utilized was the Senior Design Lab provided to us by UCF. This was the favored location of contact for many members of our group during our weekly scheduled meetings and was the default meeting place whenever the group had to discuss something urgently.

The lab itself is home to a wide array of computers and systems meant to serve any purpose senior design groups may face while completing their project. Included in this list is a wide array of Windows PC's and Mac OS devices, computers dedicated for machine learning and AI training, and even web servers for testing local projects and database structures. Our group never had to use these systems due to the scope of our project, but there have been considerations of using these systems next semester. The lab is also home to various desk stations and displays where groups can sit and work together while also presenting to one another. Our group found that this was the resource we ended up using the most. It is true that we primarily used the lab as a place to meet, but when deciding what tech stack we were going to use, our group used the external monitors to begin developing apps together to get a better idea of how each system fit our development styles. Aside from that, the senior design lab had the secondary purpose of being a location where we could store the equipment granted to us by our sponsors.

The lab is home to lockers where teams can store any equipment pertinent to their project. Since our sponsor gave us a router, multiple raspberry pi's and 3d printed components our group decided it would be important to have a secure location where this equipment would not be damaged. The need for easy access to the supplies by any group member, was also deemed vital, so we decided using a locker would be a smart decision.

11.2. 3D Printing Facilities

Aside from the senior design lab, our group did not really require any access to other third-party facilities, except for 3D printing companies and locations on campus. Our sponsor is primarily in charge of all aspects related to 3D printing add-ons, but we know he used both 3D printing facilities offered on campus, and from companies found in the Orlando area.

12 Milestone Chart

Below is a tentative timeline for the Summer 2022 semester, followed by a timeline for the Fall 2022 semester.

Milestones	Due Date
Summer 2022 Semester	
Group kick-off meeting	05/31/2022
Senior Design Boot Camp	06/06/2022
Roles Assignment	06/07/2022
Initial Meeting with Sponsor (Evan Wallace).	06/08/2022
First Check-In & Group Discussion on How to Complete the Project.	06/09/2022
Revision Meeting with Sponsor (Clarifying Ideas).	06/10/2022
First of Recurring Weekly Group Meetings	06/12/2022
Initial 15 Pages of Final Document Due.	06/19/2022
Conclude Initial Background Research.	06/21/2022
Beginning of Bi-Weekly Meetings with Sponsor (Dr. Amy Giroux).	06/22/2022
Acquired Router, Raspberry Pi & NFC Tag.	06/23/2022
Discuss Database Design and Create ERD.	06/26/2022
Establish What APIs are Needed Between Database and Web-Builder Website	06/29/2022
Turn in 15 Pages Per Group Member	07/03/2022
Request Purchase of Any Extra Needed Items	07/05/2022
Create Database and Link Database to Backend	07/07/2022

Finish Initial API's & Begin Making Website that Will Store Tours.	07/10/2022
Begin Designing Basic Template for Tour Websites.	07/14/2022
Update Documentation.	07/17/2022
Establish Connection Between Basic APIs and Web-Builder Website.	07/21/2022
Finish Template for Tour Websites	07/24/2022
Establish Connection Between Raspberry Pi and Router	07/26/2022
Testing of NFC Tags & Initial Connection to Router. Attempt to Access a Site on Raspberry Pi from a Tag.	07/27/2022
Finish All API's & Main Website, Test All User Functionalities.	07/28/2022
Conclude Revisions to Final Document.	07/31/2022
Turn in Final Document	08/01/2022
Fall 2022 Semester	
Review Plan for Semester.	08/15/2022
Design Method for Pushing Tour Websites.	08/18/2022
Begin Testing Pushing Tour Sites	08/21/2022
Final Testing of Pushing Tour Site with a Complete Example Site.	08/23/2022
Final Testing of NFC Tags.	08/30/2022
Planning of Local System at UCF for Presentation.	09/04/2022
Begin Designing Tour Site for Final Presentation	09/08/2022
Conclude Testing of Test Tour Site.	09/11/2022
Revising Final Errors.	09/18/2022
Document Revisions.	09/25/2022

13 Summary and Conclusions

In order to set up a foundation solid enough to successfully finish our project during the Fall 2022 semester, our group has worked diligently to attain a deep understanding of the problem we are trying to solve and all available solutions. We have taken the plan provided to us by our EMURR sponsor team and modified aspects as to make a robust and complete product. A great amount of effort has been dedicated to researching servers, hardware alternatives, software stacks, and even related technology to what we are currently developing. Throughout the duration of the project so far, we have established regular meeting schedules with both our sponsor and amongst ourselves, while also outlining an effective plan for the next semester.

Tasked with creating a website builder capable of generating user designed website pages that are able to be hosted on a Raspberry Pi, our team has shaped the plan provided to us, into something we believe is far more effective. Beginning with what we are not changing, we have decided to keep the original router given to us, as well as maintaining the idea of a simple website builder that allows one to upload created websites to raspberry pi connected to said router. Furthermore, users will still be able to access the broadcasted site through NFC tags in the immediate area, as well as access photos or videos embedded into these pages. What we have decided to change however, primarily begins with the Raspberry Pi being used. We will no longer be using a Pi 2 and instead will be using a Pi 3 B+ for our hosting purposes. Furthermore, we have decided to implement the partial use of third-party html generators for our page production and have landed on NEXT being our tech stack of choice. Overall, we are confident that we have made the correct adjustments to our sponsors plan and will be able to deliver a product that exceeds their expectations.

Albeit a simple concept, certain aspects of our project proved to be much more complex than we initially anticipated. Our focus for this semester was to find enough valuable information that could inform our hardware and software decisions going forward. Fortunately, we were able to achieve those goals as well as begin prototyping server options and HTML text converters. Due to this, we look forward to the beginning of the next semester, and properly implementing the plan we have laid out.

14 References

[1] “TinyMCE | Security,” *TinyMCE Documentation*. https://www.tiny.cloud/docs/advanced/security/#cross-sitescriptingxss (accessed Aug. 01, 2022).

[2] “EU - Data Protection Overview,” *DataGuidance*, Sep. 09, 2021. https://www.dataguidance.com/notes/eu-data-protection-overview (accessed Aug. 01, 2022).

[3] “Guidelines 05/2020 on consent under Regulation 2016/679.” May 04, 2020. [Online]. Available: https://edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_202005_consent_en.pdf

[4] “California Consumer Privacy Act (CCPA),” *State of California - Department of Justice - Office of the Attorney General*, Oct. 15, 2018. https://oag.ca.gov/privacy/ccpa (accessed Aug. 01, 2022).

[5] “Wikipedia:Protection policy,” *Wikipedia*. Jul. 21, 2022. Accessed: Jul. 31, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Protection_policy&oldid=1099515682

[6] “Terms of Service.” Accessed: Aug. 01, 2022. [Online]. Available: https://www.youtube.com/static?template=terms

[7] “17 U.S. Code § 107 - Limitations on exclusive rights: Fair use,” *LII / Legal Information Institute*. https://www.law.cornell.edu/uscode/text/17/107 (accessed Aug. 01, 2022).

[8] “Raspberry Pi Foundation – About us,” *Raspberry Pi*. https://www.raspberrypi.org/about/ (accessed Jun. 29, 2022).

[9] M. Zolnierczyk, “Raspberry Pi network speed test: RPI2, RPI3, Zero, ZeroW (LAN&WiFi),” *NotEnoughTech*, Apr. 05, 2017. https://notenoughtech.com/raspberry-pi/raspberry-pi-internet-speed/ (accessed Jul. 02, 2022).

[10] “2.2 Dissecting a Network Packet.” http://books.gigatux.nl/mirror/snortids/0596006616/snortids-CHP-2-SECT-2.html (accessed Aug. 01, 2022).

[11] “AC1200 Wireless Dual Band Gigabit Router.” https://www.tp-link.com/us/home-networking/wifi-router/archer-c1200/ (accessed Aug. 01, 2022).

[12] “Learn What Are NFC Tags — A Beginner’s Guide,” *Nomtek*. https://www.nomtek.com/blog/what-are-nfc-tags (accessed Aug. 01, 2022).

[13] “NTAG213/215/216 Product data sheet.” NXP Semiconductors. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/NTAG213_215_216.pdf

[14] MaryQiu1987, “Define relationships between tables in an Access database - Office.” https://docs.microsoft.com/en-us/office/troubleshoot/access/define-table-relationships (accessed Jul. 01, 2022).

[15] “Most popular relational DBMS 2022,” *Statista*. https://www.statista.com/statistics/1131568/worldwide-popularity-ranking-relational-database-management-systems/ (accessed Jul. 01, 2022).

[16] “What is a Relational Database? – Amazon Web Services (AWS),” *Amazon Web Services, Inc.* https://aws.amazon.com/relational-database/ (accessed Jun. 30, 2022).

[17] Z. Tejada, “Non-relational data and NoSQL - Azure Architecture Center.” https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data (accessed Jul. 02, 2022).

[18] “Sharding — MongoDB Manual.” https://www.mongodb.com/docs/manual/sharding/ (accessed Jul. 02, 2022).

[19] “MySQL :: MySQL Workbench.” https://www.mysql.com/products/workbench/ (accessed Jul. 29, 2022).

[20] “What is AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/what-is-aws/ (accessed Jul. 30, 2022).

[21] E. B. Setiawan, A. Setiyadi, and R. Wahdiniwaty, “Quality Analysis of Mobile Web Server,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 662, no. 2, p. 022043, Nov. 2019, doi: 10.1088/1757-899X/662/2/022043.

[22] T. Simunic and V. Deolalikar, “Server Controlled Power Management for Wireless Portable Devices,” *ResearchGate*, [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.877.8967&rep=rep1&type=pdf

[23] K. R. Gsangaya, S. S. H. Hajjaj, M. T. H. Sultan, and L. S. Hua, “Portable, wireless, and effective internet of things-based sensors for precision agriculture,” *Int. J. Environ. Sci. Technol.*, vol. 17, no. 9, pp. 3901–3916, Sep. 2020, doi: 10.1007/s13762-020-02737-6.

[24] R. T. Tse and Y. Xiao, “A portable Wireless Sensor Network system for real-time environmental monitoring,” in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2016, pp. 1–6. doi: 10.1109/WoWMoM.2016.7523588.

[25] M. U. H. A. Rasyid, F. A. Saputra, and A. Prasetiyo, “I-ON Smart Controller: Portable Smart Home Solution Based on Arduino And Raspberry Pi,” in *2018 International Conference on Applied Science and Technology (iCAST)*, Oct. 2018, pp. 161–164. doi: 10.1109/iCAST1.2018.8751609.

[26] R. Dhuny and N. A. Mohamudally, “RPI64Box: A portable 3-tiered LAMP stack in a 64-bit Operating System environment,” *Softw. Impacts*, p. 100390, Jul. 2022, doi: 10.1016/j.simpa.2022.100390.

[27] G. Birajdar, “Implementation of Embedded Web Server Based on ARM11 and Linux using Raspberry PI,” *Int. J. Recent Technol. Eng. IJRTE*, vol. 3, no. 3, [Online]. Available: https://www.ijrte.org/wp-content/uploads/papers/v3i3/E0892112513.pdf

[28] L. Nagy and A. Coleșa, “Router-based IoT Security using Raspberry Pi,” in *2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, Oct. 2019, pp. 1–6. doi: 10.1109/ROEDUNET.2019.8909551.

[29] “Storybook: UI component explorer for frontend developers.” https://storybook.js.org/ (accessed Aug. 01, 2022).

[30] SyntaxTerror, *English: A comparison of the visible color spectrum in common types of color blindness.* 2022. Accessed: Aug. 01, 2022. [Online]. Available: https://commons.wikimedia.org/wiki/File:Color_blindness.svg

[31] “A successful Git branching model,” *nvie.com*. http://nvie.com/posts/a-successful-git-branching-model/ (accessed Aug. 01, 2022).

[32] “Trunk-based Development vs. Git Flow,” *Toptal Engineering Blog*. https://www.toptal.com/software/trunk-based-development-git-flow (accessed Aug. 01, 2022).

[33] “Faculty and Staff,” *Texts and Technology Ph.D.* https://cah.ucf.edu/textstech/faculty-staff/ (accessed Jul. 28, 2022).

[34] “Evan Wallace - Center for Humanities and Digital Research.” https://chdr.cah.ucf.edu/gra/evan/index.html (accessed Jul. 28, 2022).

[35] “Norris, Sarah,” *UCF Libraries*. https://library.ucf.edu/staff/norris-sarah/ (accessed Jul. 31, 2022).