ROOTS: A SUBSCRIPTION PLATFORM FOR DIGITAL MEDIA

BY
HAYLEE MILLAR

A Thesis

Submitted to the Division of Natural Sciences
New College of Florida
in partial fulfillment of the requirements for the degree
Bachelor of Arts
Under the sponsorship of Matthew Lepinski

Sarasota, Florida
May, 2021

Acknowledgments

There was no guarantee when I started my journey in computer science I would enjoy it as much as I have. I would like to first and foremost thank Dr. Matthew Lepinski, who has endlessly encouraged me and helped battle my Imposter Syndrome; without you, I would have never taken the plunge! I am forever grateful I met you in the Fall of my senior year in high school.

Secondly, I would like to thank my family for their support through my entire college education. Your foundation and dependability have kept me sane. Thank you for reminding me I am a strong, capable human.

I would also like to acknowledge Dr. Tania Roy, who has been my professor, supervisor (as her teaching assistant), and friend. Your encouragement and care do not go unnoticed — thank you.

Finally, I would like to show appreciation to my professors and friends. You have made my career at New College oh-so-memorable. I have learned so much from you all and will take those lessons into the future with me.

Table of Contents

ROOTS: A SUBSCRIPTION PLATFORM FOR DIGITAL MEDIA

Haylee Millar

New College of Florida, 2021

ABSTRACT

Poorly-targeted advertisements have plagued the Internet for some time now, which has given way for the subscription model to rise in popularity. Advertisements (ads) have the potential to irritate and deter users, as they are vying for the user's attention alongside a website's content. Using a subscription model, users are no longer distracted from the creator's content with ads, thus increasing user happiness.

Roots is a subscription platform where content creators can monetize their media websites. Users can explore and subscribe to new websites, as well as manage their subscriptions. Creators can create landing pages describing their content site as well as subscription tiers with associated benefits.

Such design & behavior is achieved through using free and open-source software Linux, Nginx, MySQL, and PHP. Using the PHP framework, Symfony, and JSON Web Tokens, Roots sends authentication data to content creator sites alongside an inbound consumer when they desire to leave Roots.

The codebase is managed and made publicly available in a GitHub repository (https://github.com/hayleemillar/thesis-project).

Matthew Lepinski

Division of Natural Sciences

Introduction

Since the birth of the World Wide Web in 1993, digital media has been a part of the Internet landscape. Digital media can be defined as a form of information or entertainment aimed to be consumed by others through technological means like computers, tablets, and cellphones. Traditional analog media like radios and magazines made room for the Internet's entertainment debut. Media has now transformed to include audio, video, social media, advertising, news, literature, as well as others in both digital and traditional spaces [9].

Companies and individuals quite often seek to monetize their content websites. The first step before monetization is to secure an audience for the content. This is most often easily achieved through social media platforms and public relations support. For example, *The New York Times* is a subscription news platform where they attempt to reach new audiences through Instagram [13] and Twitter [12] in addition to their website [11].

Once an audience has been established, the next step is to choose a form of monetization. The following section will argue for a subscription model, where users are free from distracting ads.

Roots is a subscription platform where creators can monetize their content, and consumer users can manage all of their subscriptions in one place. Creators can create landing pages for their content's websites with subscription tiers available. Users can subscribe to websites by selecting subscription tiers, where they can then manage all active

subscriptions in their account profile. Roots provides exposure and monetization for creators, while the consumers can manage all subscriptions to independently hosted content websites on one singular platform.

Roots will be described in the chapters following the background. Its design and architecture will be outlined, which relies heavily on open source software. A glossary has been included at the end of the thesis to assist readers with understanding. All code referred to can be found on GitHub (https://github.com/hayleemillar/thesis-project).

Chapter One: Background

There is an abundance of free media available on the Internet. Thus, creators of digital media must compete with a very low reference for the price point of comparison if they wish to monetize their content [7]. In such a free-centric environment creators are left with two main models: advertisement and subscription.

Online advertising, which made up 41% of *all* advertising in 2017, has become a very common part of the Internet experience. Their appeal is high in the fact users can consume free content in return for dealing with advertisements being present on the web page. The user's attention is the commodity; when websites launch ad campaigns, their choice of ads is then displayed to a target set of users throughout a website's content on other participating websites. The very nature of sprinkling ads within and besides content competes against the content for the user's attention. The ad could be interesting to the user, or it could be irritating, which is still nonetheless decidedly distracting. Ads are

usually not random, as the goal is to target users who may be interested in the service or product, but they can be repetitive. It is not uncommon to use repetition as a tactic for capturing the user's attention, which can degrade the user's experience if the ad is of no interest to them [25].

Another option is a subscription-based model, where users pay some recurring fee in return for online content. The user's whole attention is on the media without any ad distractions. User experience becomes improved while maintaining a viable business model. However, as previously mentioned, consumers of online content are often unwilling to pay for content due to vast amounts of free media online [7]. However, platforms like Patreon and websites like *The New York Times* are both examples of a subscription model working positively within a business model, which indicates users are open to shifting into accepting subscriptions.

The subscription-based model is the focus of this thesis. Advertisements have become more and more ineffective as time has progressed. Using traditional practices to place ads within the content, the ads can become intrusive and degrading to the user experience. The problem not only lies within the ad model itself, though; Internet users have begun using 3rd party ad blockers to personally improve their experience. Such use of blockers leads to a loss in income through ads, as they are no longer being seen or clicked on by users. Without user clicks, the advertisements are not fulfilling their role in creating a monetary return. Considering the drawbacks and resistance against using ads, Roots will use a subscription model for its users.

Popular websites like Patreon, Memberful, and Podia exist on the Internet as options for creator monetization of digital content through a subscription model. Patreon is a platform that has accrued over 100,000 content creators. Each creator has a page on Patreon where they post content like videos, podcasts, and blogs. Patreon has three tiers for creators to choose from. The lowest tier is a 5% transaction fee plus payment processing, which offers a landing page for the creator, communication (email, direct messaging), and free workshop videos. The middle tier is an 8% transaction fee plus payment processing, which offers the above plus membership tiers, analytics, the ability to offer limited-time deals, live-streamed workshops from creators, integration of other apps (like Facebook, for example), and priority customer support. The top tier of Patreon is a 12% transaction fee plus payment processing, which offers the above plus a dedicated partner manager (a Patreon employee), the ability to sell merchandise, and multiple accounts for the ability to accommodate a team [16].

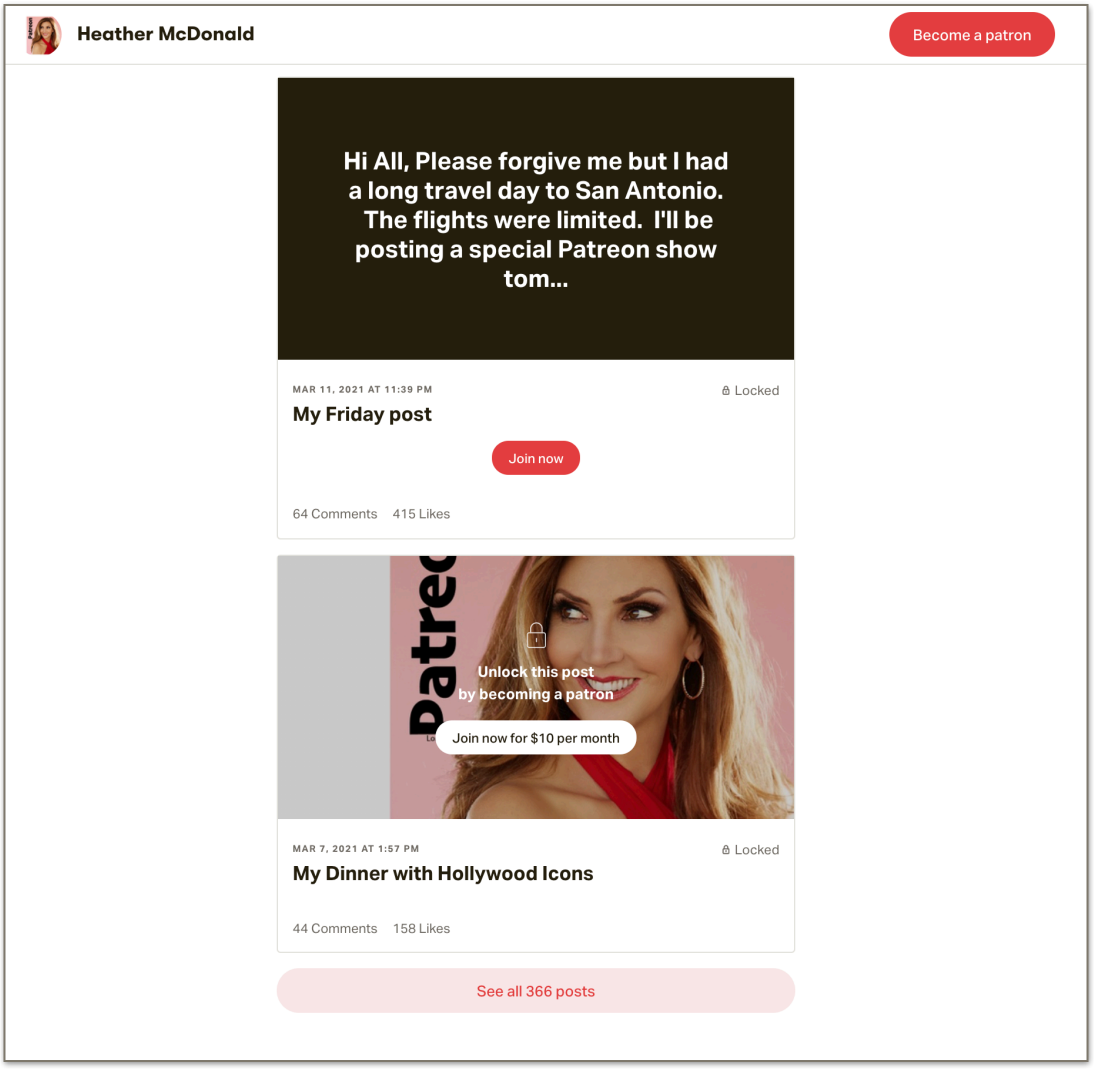


Figure 1: Example Patreon landing page feed. The button "See all 366 posts" at the button calls the user to action. Source: https://www.patreon.com/juicyscoop

Patreon bears a few issues as a platform in the scope of the goals of this paper. The first is the restriction of creators from presenting their content the exact way they please. Every creators' page is white with the same minimal accents. Oftentimes it is difficult to distinguish between posts because of the lack of color contrast. Second, the creators' posts may only be viewed in a linear format, as each post is contained in a card. While

this linear layout may be filtered in different ways, the ability to filter is only by clicking a button with the label "See all *[total number posts]*". Only then can users more narrowly define their results based on the media type, subscription tier, and month posted. If creators had more control, they could create designs they feel may be more suited to their content.

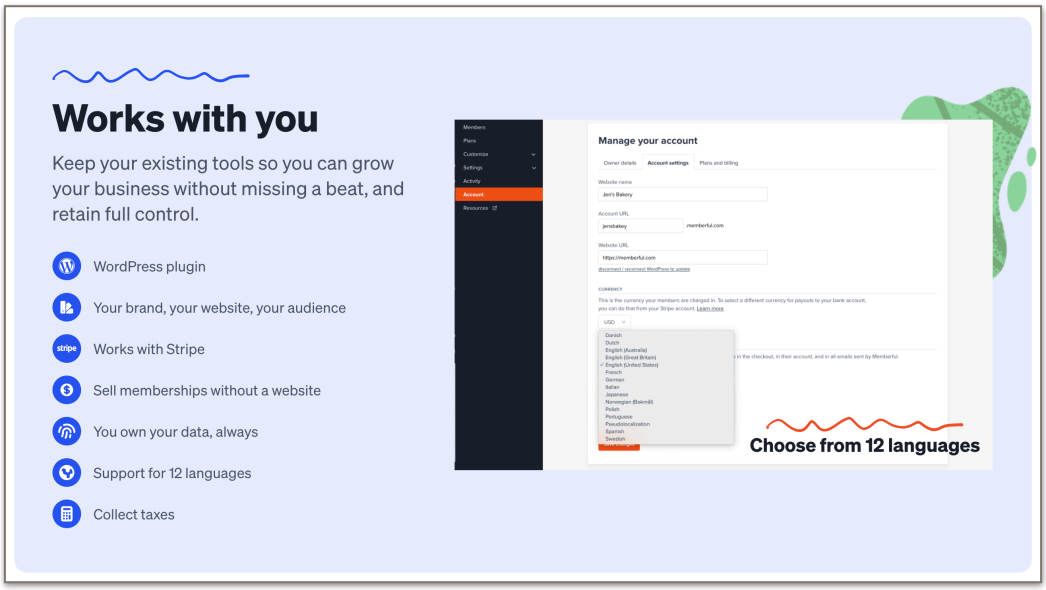


Figure 2: Memberful Overview. Source: https://memberful.com/how-it-works/.

Memberful, a Patreon company, provides a technology that integrates into the creator's website and has three different pricing plans. The Starter plan ($0/month + 10% transaction fee) allows creators to have 2 plans (essentially, websites), and free website integrations. The Pro plan ($25/month + 4.9% transaction fee) allows creators have unlimited plans (websites), all service integrations (Mailchimp, Discourse, etc.), the ability to have coupon codes, multiple accounts to accommodate team members, enable a free member tier, analytics, API+ webhooks, and custom branding. The Premium plan

($200/month + 4.9% transaction fee) allows creators to have all of the above plus the ability to sell group subscriptions (for example, to an organization of sorts like a school) and the removal of all Memberful branding [10].

Memberful is the closest technology to the one to be proposed, but it misses the ease of user experience on the Internet on a larger collaborative scale. If users had one platform to manage their subscriptions to their favorite content creators' websites, as Patreon does with their creator pages, then they can consume content more easily, thus enhancing the user experience. Users might be more willing to spend their money and contribute to creators if they can have an overview of their overall subscription spending. They may also be able to explore further for more creators.

Podia is a platform allowing websites to be monetized through subscriptions but requires the website to be hosted within their architecture. There are over 25,000 content creators that use Podia. It has two different pricing plans. The Mover ($39/month) allows creators to have a page, communication (email, direct messaging), unlimited hosting, free migrations, 7 days/week customer service, and online courses/downloads for additional help. The Shaker ($79/month) allows creators to have in addition to the above-mentioned the ability to implement membership subscriptions, a blog, embedded checkout, affiliate marketing, and the ability to use 3rd party code. Podia differs from Patreon and Memberful in the way it has no transaction fees. The downfall to Podia is the creator cannot choose where their website is hosted. Also, users have no central platform to explore for new creators, isolating each website to their respective audience [17].

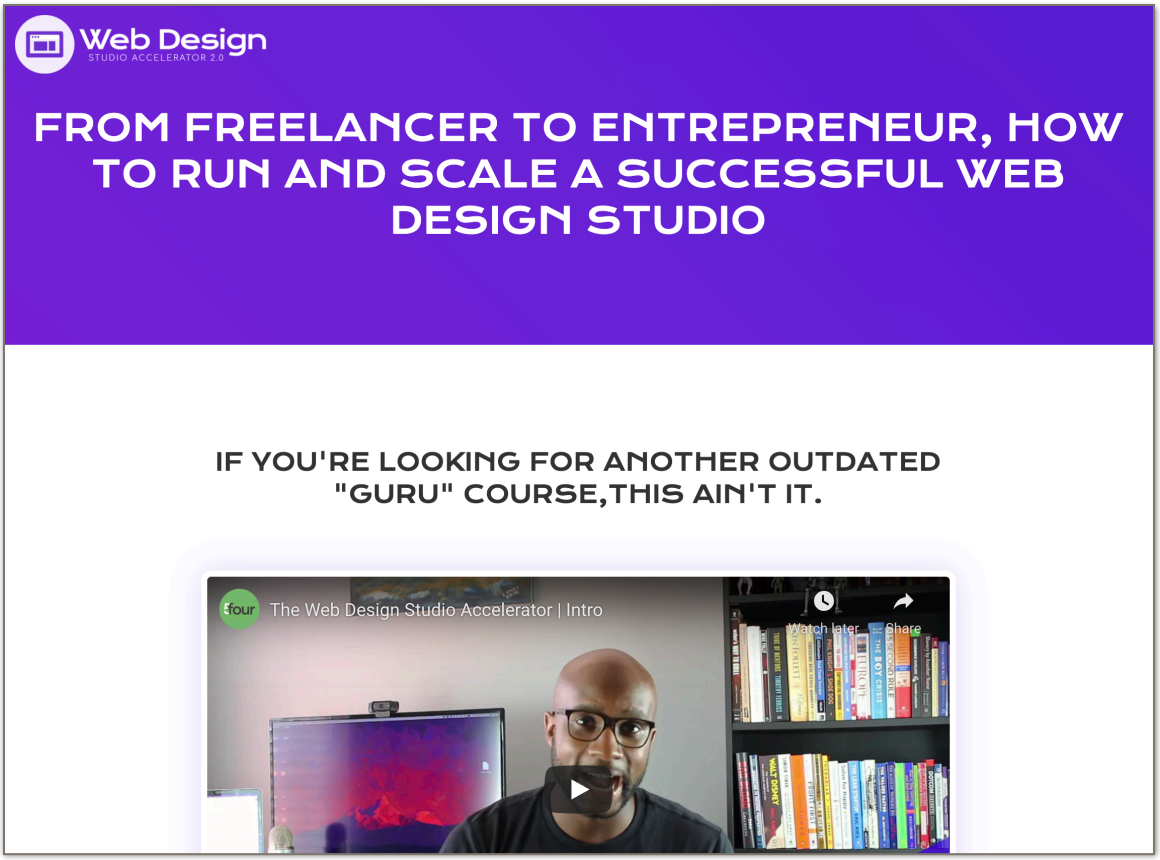


Figure 3: Example Podia website, Web Design Studio Accelerator 2.0. Source: https://www.johndsaunders.co/courses/web-design-studio-accelerator.

No current platform makes the subscription model easy for the media consumer while also keeping the media ownership rights with the creator. Roots will include a central website for users to manage their grouping of subscriptions, where they may also explore new creators. This allows creators with similar content to share audiences and benefit from a social platform structure. Users can also be more at ease knowing their payment for multiple subscriptions is held and billed under one company. If a user is on a Premium Memberful website, there is no Memberful branding. The user may not fully be aware of who has their payment information. If users had a central place to manage

subscriptions, the ambiguity would be lifted. As for the content creators, Roots appeals to those wishing for full autonomy over their content and its presentation. It should also appeal to some creators for them to share and gain users through social aspects of the platform.

Chapter Two: Platform Design

The proposed platform, Roots, provides a service mutually beneficial to the content creator and the content consumer. Creators can create landing pages describing their website and its content. Subscription tiers are used as the monetization format, allowing creators to define the tier price and provide a brief description of what users can expect. Consumer users may explore landing pages and select tiers to gain access to the content on the creator's content site. Users are also able to view all of their subscriptions at once via the user profile page, allowing for easy management and comparison.

The overall design goal of Roots is to maintain simplicity. While the site's design is not the main focus of this paper, it is important to highlight key characteristics within the platform to establish a baseline of understanding. The remaining portion of this chapter will provide a high-level overview of the design choices made for Roots as a platform.

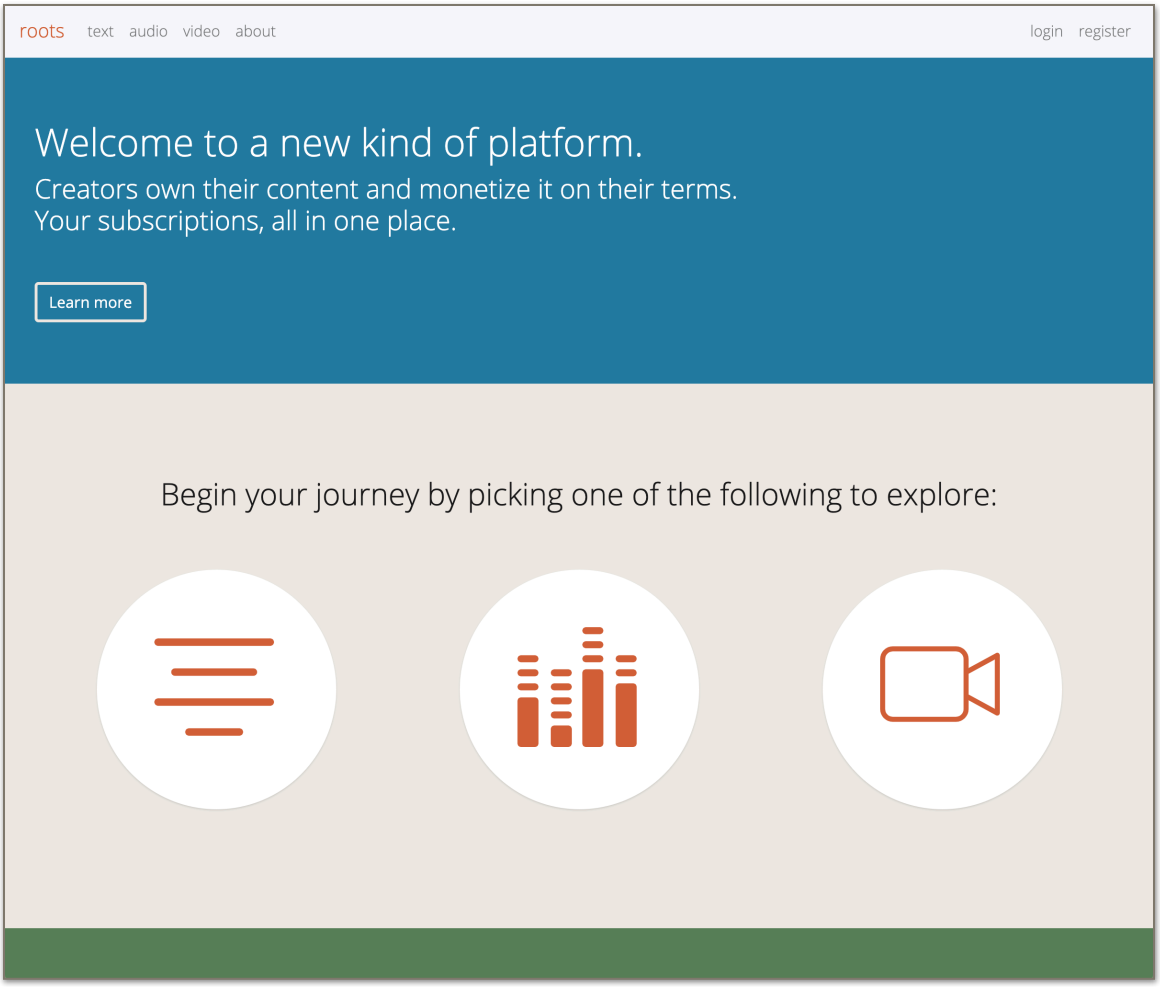


Figure 4: Roots homepage.

First and foremost, the homepage is important as it is the first page many users will most likely see first before navigating to the remainder of the website. Its desired effect is to be an interactive online brochure. The homepage is a quick way for users to understand what Roots is and why they may be interested in its purpose. The platform has a target audience of those who consume online content; it is always important to remind the audience what product is being sold and why they should be interested in trying it. What can they do with Roots? Could it be profitable? Is it educational or entertaining? Such information should be easy and readily available upon reaching the homepage of any

website advertising a product or experience. Thus, the first text on the Roots homepage is notifying users they can both monetize content (if they have any) and manage subscriptions. A call to action (CTA) is employed to allow the user to get more information about Roots by providing a "Learn more" button underneath the first text on the homepage. There is also a CTA for users to explore the content creators available via the platform. If they feel confused, the CTAs will make it easy for users to find the next logical step. Finally, reviews of Roots are provided on the homepage to inspire ethos and credibility within potential new users. If others enjoy Roots, maybe they will too [18].

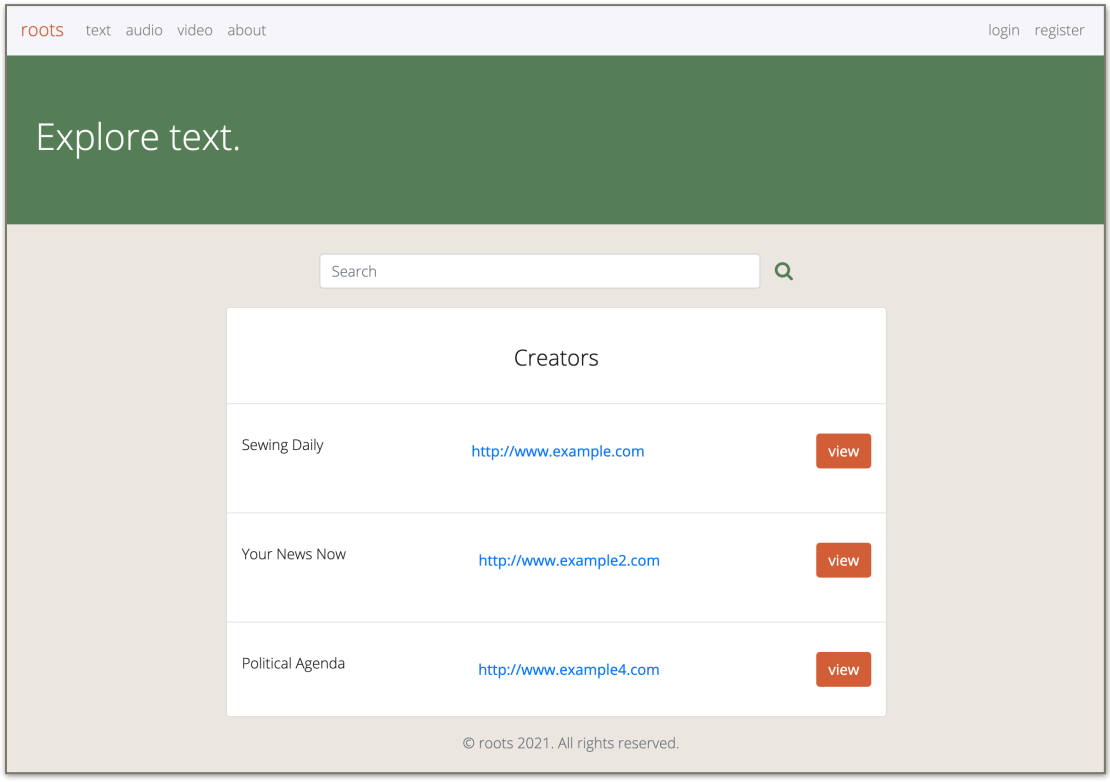


Figure 5: Example Roots Text Explore Page.

Content creators on Roots fall into three main categories: text, audio, and video. Each category does not need to be explicitly *just* text or *just* audio; they are merely buckets for users to explore creators. Thus, creators need to label their websites as best they can for optimal discovery by users. Each category is listed and available to click within the main menu navigation at the top of every page. Users must be able to easily explore and find new content they find interesting, otherwise, Roots has lost meaningful functionality.

The explore pages for each content category follow the same structure; users can choose to either search using keywords or scroll through trending creators. Sometimes, a user may be looking broadly for a new creator, making a browse through the trending creators the easiest way to achieve the desired results. On the other hand, some users may desire to find a specific creator or search for a specific category (say, maybe sewing). Then, using the search bar, the user can refine their results. Either way, the explore pages are set up with both functionalities in mind and easy for the user to navigate. The page results will always provide the user with a content website's title, the URL, and a button to view its landing page if they desire to learn more.

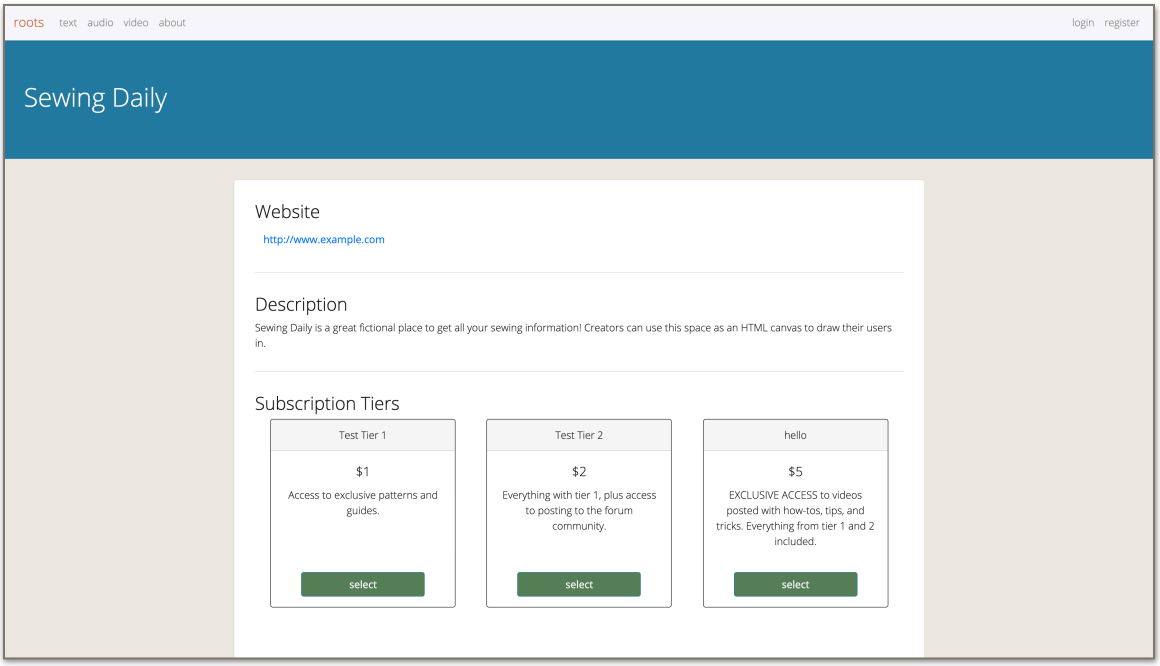


Figure 6: Example Roots Landing Page.

Landing pages for content websites can be created by any user. Each page has a title, a website URL, a description, and the subscription tiers users can choose. The title should match or have some relation to the website advertised. The website URL will need to be valid as it will become a clickable link within the landing page for users to click, whether or not they are subscribed. Creators can save the page description as HTML text, giving them more creative freedom to advertise their website as they please. Finally, the subscription tiers have titles, amounts (in dollars), and text descriptions. Users can select *one* subscription tier of their choosing for each landing page. They are also able to leave reviews to inform others on the website's content.

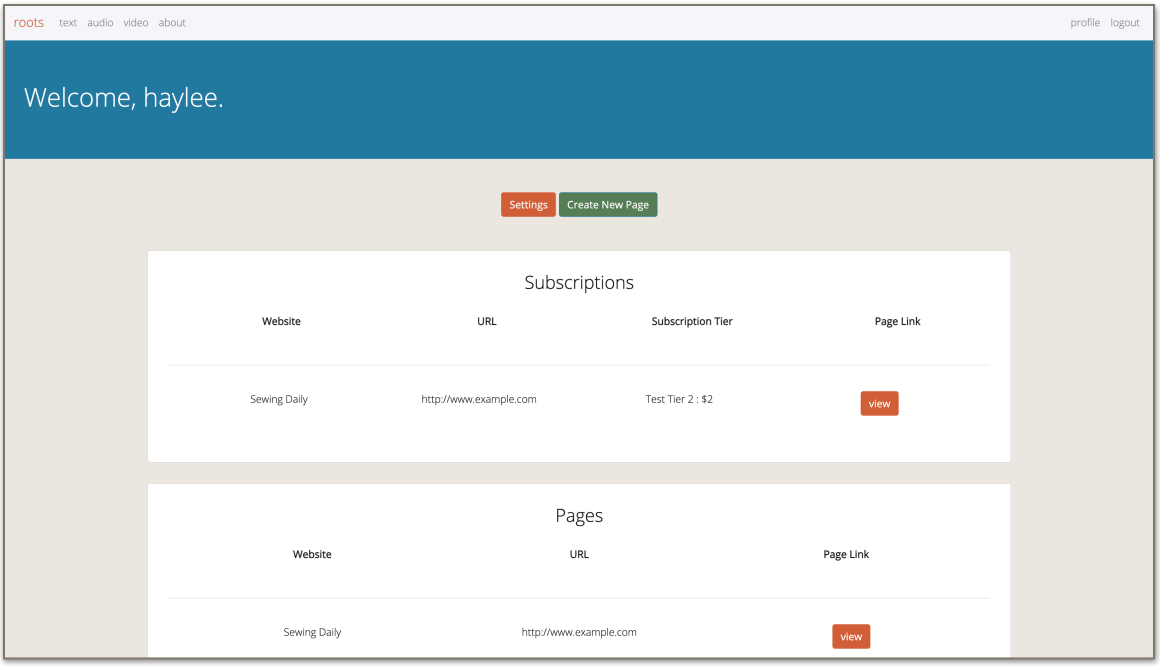


Figure 7: Example Roots Profile Page.

Finally, when logged in every user has a profile and settings page. On the profile page, users can view both their aggregate subscriptions and landing pages they own in their separate respective tables. The website title, URL, as well as a button to the landing page, are available for immediate viewing in both. Users can also see which tier they have selected within the subscriptions table.

Before both tables are displayed, there are two buttons at the top of the page. One sends the user to their settings page; the other allows the creation of a new landing page. The settings page allows users to change their name, email address, and password. The form for creating a new landing page allows users to specify the URL, title, type, description, and whether the page should be private. Tiers may be added after the page has been created.

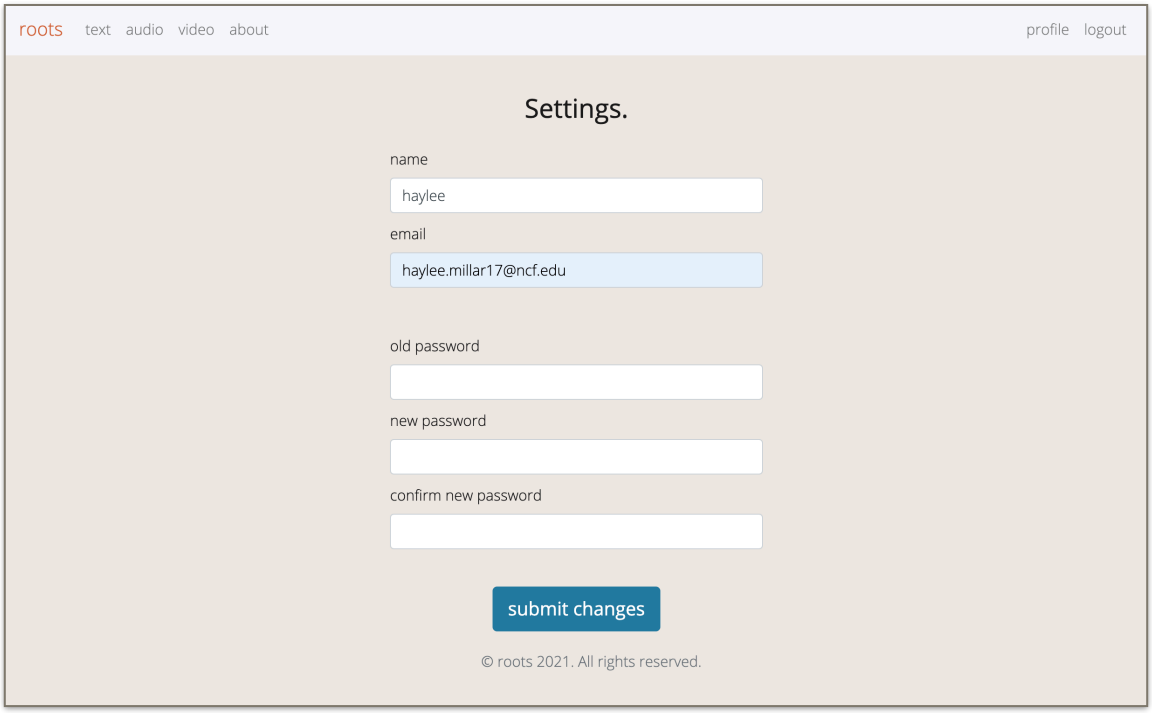


Figure 8: Example Roots User Settings Page.

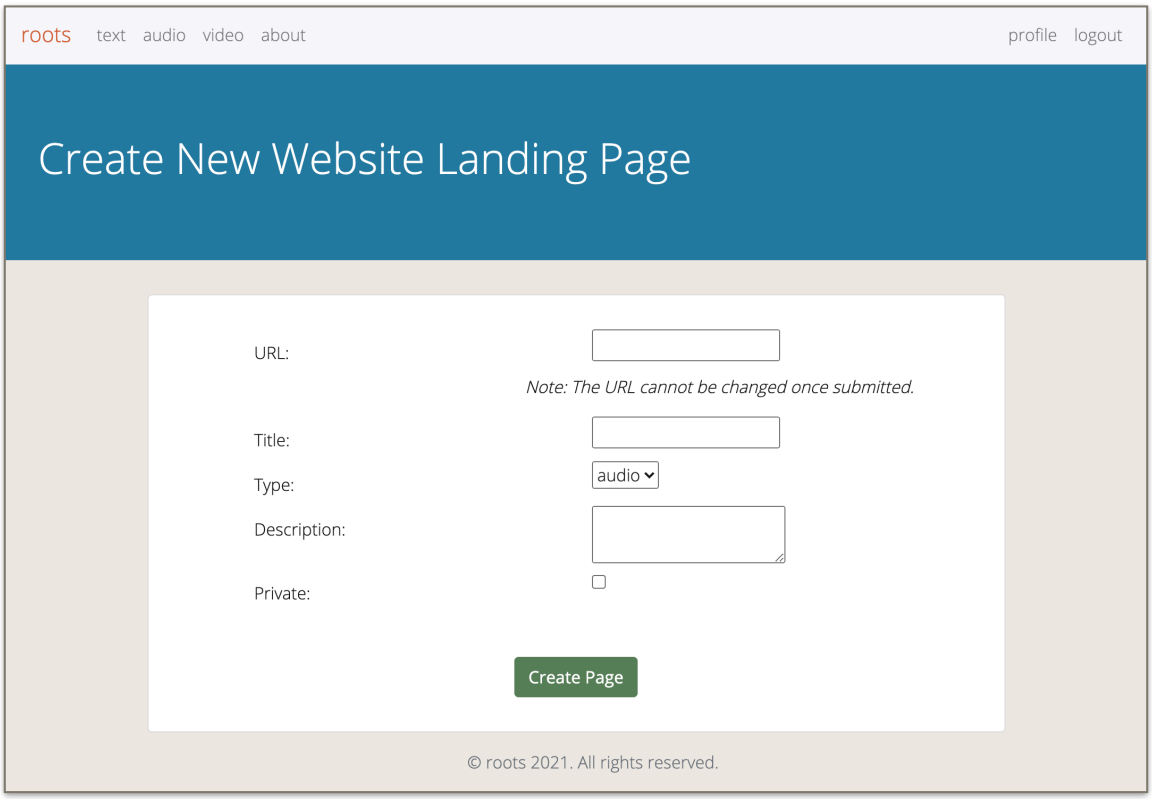


Figure 9: Roots Create Landing Page.

Chapter 3: Platform Architecture

The technology stack on which Roots is built is known as the LEMP stack: Linux, Nginx, MySQL, and PHP. Each technology in the stack shares an important characteristic-- they are all open source. The codebase is made publicly available for other people to use, maintain, and build upon, usually via a GitHub repository. This is juxtaposed to closed source software, which is maintained by a company or team of people [15].

There are many reasons to use free and open source software (FOSS), and why it is often a popular option for web development. First and foremost, FOSS removes the barrier of having to reinvent the building blocks for a good website application. Providing a free baseline for all people allows for more difficult problems to be solved. Second, the total cost of ownership (TCO) for FOSS is very low. There are no fees or payments required to use the software. There are also forums and communities online, like Stack Overflow [19], where developers can seek support for their issues. While there are oftentimes courses and camps developers can pay to learn how to use a specific FOSS better, the initial requirements are minimized by the communities and readily available information found online. Finally, Linus's Law states that "given enough eyeballs, all bugs are shallow". Bugs are an inevitable part of software development, as humans are prone to err. There are developers around the world maintaining and furthering FOSS. Bugs are more easily and quickly fixed when more than one person is searching for the answer, each of them with a different set of expertise. Not only beneficial in supporting

maintenance work, but the international influence also brings different viewpoints and considerations a homogenous team might otherwise overlook [15].

Linux has grown to be considered the industry standard for server operating systems (OS) for many reasons. First, it is available on more platforms than any other OS. Linux also is known for its ability to be used in a wide variety of applications. If a developer is not sure exactly the architecture of their application, Linux provides flexibility in its structure. At its core is stability and security that comes naturally from an open source software as highlighted in the previous paragraph. All of these characteristics make Linux the optimal choice for the proposed platform, especially given its experimental nature. No matter the final form, the operating system will be able to handle the changes in architecture and other alterations during the development phase of Roots [4].

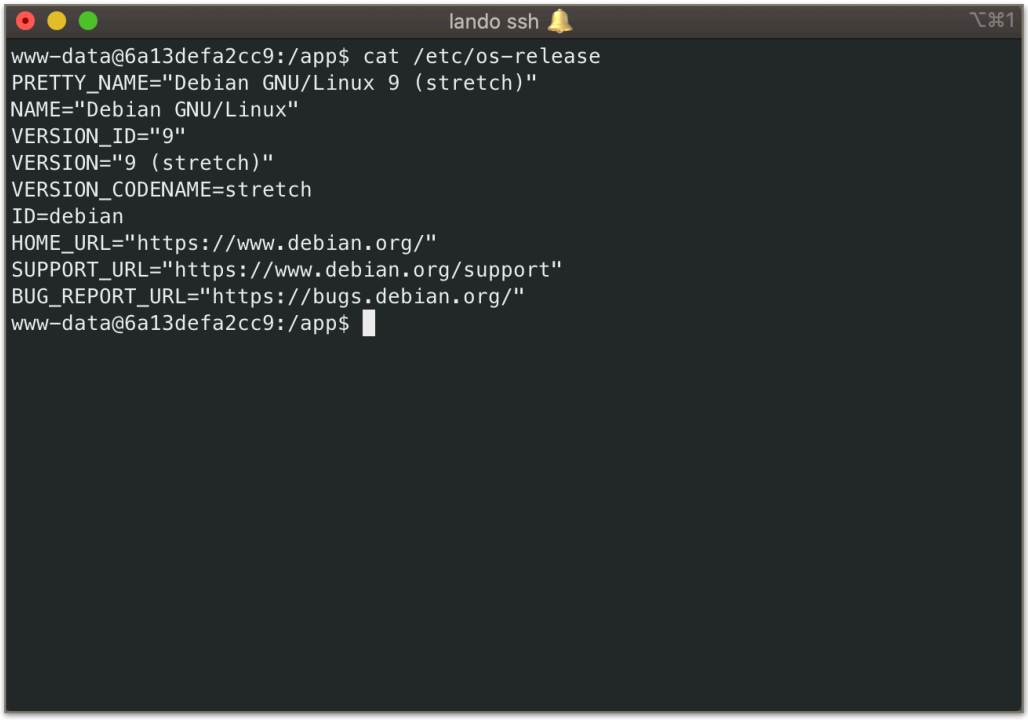


Figure 10: Roots Linux OS details via Terminal.

Nginx is powerful web serving software that can easily handle a varying number of requests. Released in 2004, Nginx is a service that receives all requests to a server and routes them to the proper directories (file folders). Compare Nginx to a person who directs traffic for a busy parking lot-- they are directing each car to their proper destination. Nginx is no different, except it can redirect many cars (or users) at once [14].

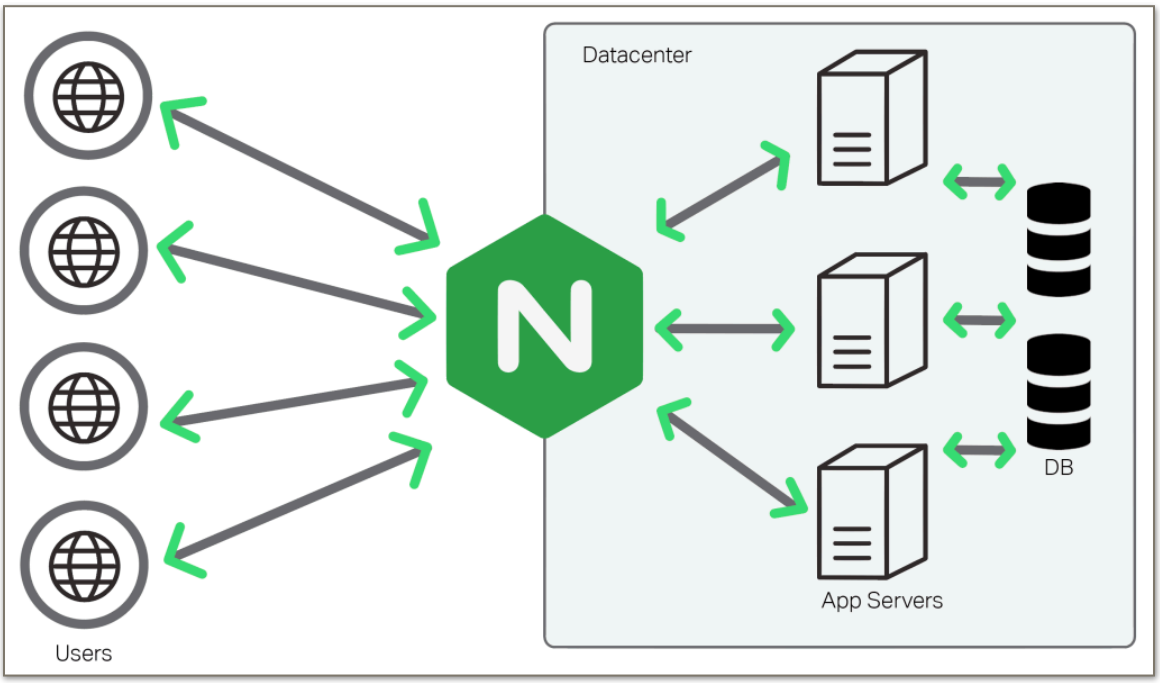


Figure 11: Nginx request handling diagram. Source: https://medium.com/@samual.r.moot/nginx-web-server-and-reverse-proxy-server-c25f68a61d92

While scalability may not be an immediate concern during development, Nginx is also important when considering the platform's future. Its load balancing capabilities will become important when Roots is in production and being visited by real users. Nginx's lightweight and easy-to-use functionality are two reasons why developers and IT professionals choose Nginx over Apache, its competitor software [14]. Nginx is chosen in

favor for Roots due to having existing experience using the software as opposed to Apache.

Roots will need to organize its data reasonably and logically within a database. A relational database allows the data to be organized into logical tables (akin to data spreadsheets), which then can be linked together using common data points among them. The ability to connect tables eliminates any need for redundant data in multiple tables, reducing complexity and making data management easier. Using the Structured Query Language (SQL), queries can be constructed to retrieve data from tables in the database. SQL is the most widely used relational database language by both developers. MySQL is the most commonly used management system for relational databases that use SQL and will be leveraged due to existing domain knowledge on the subject in contrast to other available database management systems like PostgreSQL and Db2 [8].

PHP is a great server-side scripting language that has been around since 1994. It is boasted for its three main strengths: server-side scripting, command-line scripting, and desktop applications. All notable operating systems can run PHP; it can process and manage many types of files and databases, making its flexibility highly beneficial to Roots. Whether it’s HTML, MySQL, or XML, PHP is equipped to handle it. While often criticized for its slow nature, PHP has currently been optimized and is still used on today's web. Despite many developers’ disdain over the use of PHP, it remains useful and still has great community support [24].

Symfony5 is the PHP Framework of choice for Roots. Built on the idea of using PHP components to construct your website application, Symfony has a large community of developers online to support and develop contributing packages. It provides the tools needed to be able to do more with less. Symfony relies heavily on its community to provide public modules that enhance and accelerate development. There is no need to "reinvent the wheel" and Symfony does so by providing components that have been tested and are proven to work [23].

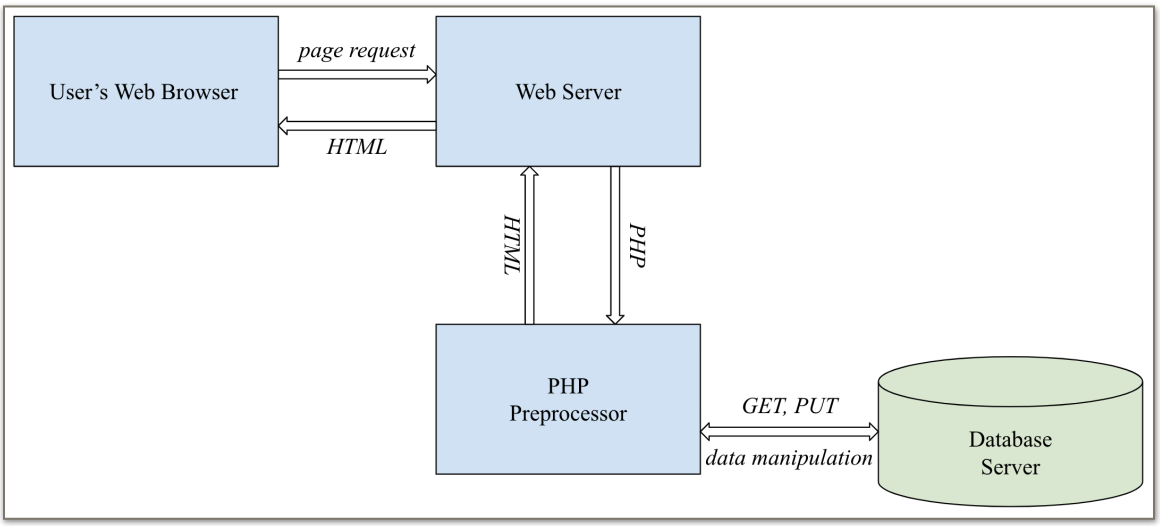


Figure 12: High-level overview of application functionality.

Some developers choose different languages and frameworks for their projects, like Django for Python. The driving force behind Django is its efficiency and ease of use due to simplified code. The high-level aspect of Django and Python allows for less code to be written than a typical PHP application. However, such frameworks take a lot of setup and are very specific in their configuration. Existing familiarity with PHP and Symfony setup

is what makes it the stronger candidate, allowing for the project to take any shape it needs.

Roots depends strongly on the database architecture being logical, as it affects both development time and application response times. The website is constantly requiring data from the database, making it important for queries to return quickly as page load times are directly related.

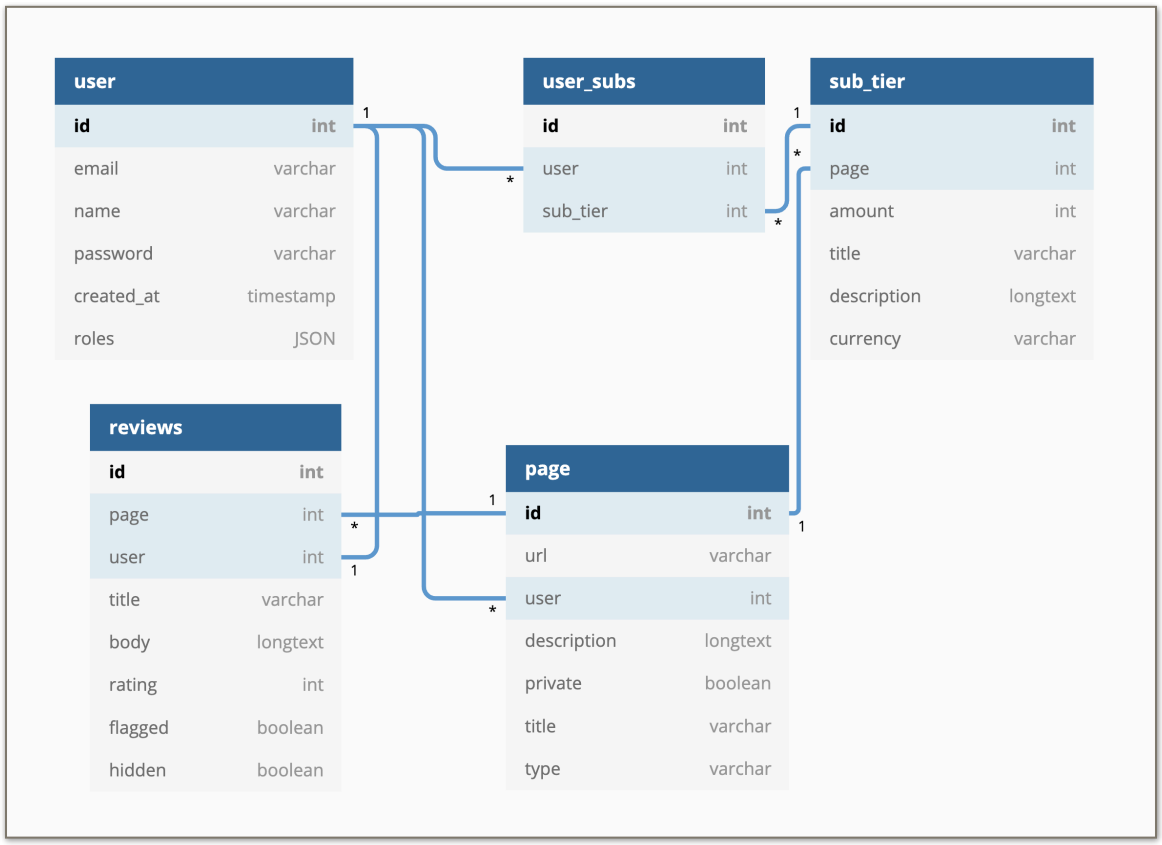


Figure 13: Entity relationship diagram of Roots database.

The following tables are the foundation of Roots: *users*, *user_subs*, *reviews*, *pages*, and *sub_tiers*. Each user has an *ID, email, name, password, creation date, and roles.* Users may change their email, name, and password as they please within the settings page of

Roots. The *roles* field is used for access control management in determining which pages users can view.

A user may have multiple subscriptions in table *user_subs*, where a subscription has the fields *ID, user* (*User ID),* and *sub_tier.* The *Users ID* field has a one-to-many relationship with *user_subs*. The *sub_tier* field will track which page the subscription belongs to.

Users may also have multiple website pages in table *pages*, where a page has the fields *ID, URL, user, title, description, and private*. The *User ID* field has a one-to-many relationship with the *pages user* field. The fields *title* and *description* are used to relay to users what type of content the user is to expect. The *private* field allows users to identify they would no longer like the page to be viewable by any other users except themselves. The *Pages ID* field has a one-to-many relationship with the *reviews* and *sub_tiers page* fields. Each page can have multiple tiers for users to choose from, as well as multiple reviews from multiple users.

The *reviews* table has the fields *ID, page, user, title, body, rating, flagged,* and *hidden*. Users can leave reviews for other users to read when deciding whether they desire to subscribe to a new content website. The review consists of a title, a body, and an integer rating that does not exceed five. A review may also be flagged for being inappropriate or hidden by the page owner.

The *sub_tiers* table has fields *ID, page, amount, name,* and *description.* The *sub_tiers ID* field has a one-to-many relationship with the *user_subs sub_tier* field, as multiple users

can select the exact tier. Subscription tiers must be related to a page, where a page may have many tiers. The *amount, name,* and *description* fields are all used to inform the user on how much the tier costs and what they are to expect if they choose it.

Chapter 4: Technology Application

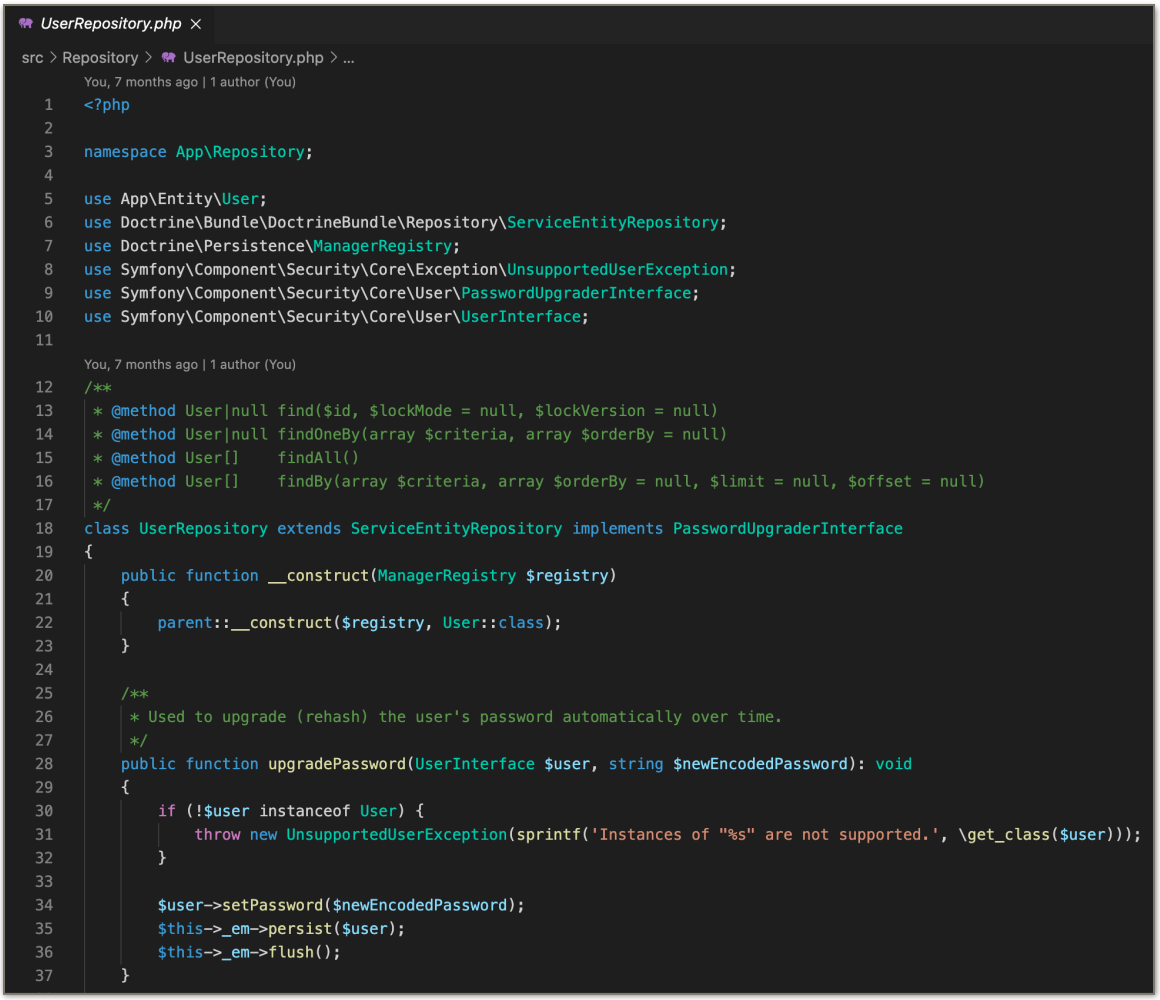
To be able to interface the database with the Symfony framework, object-relational mapping (ORM) is employed. While the MySQL service is running and available to be queried, it is not immediately available for use by other services. First, a connection must be made programmatically to the database. ORM is the mechanism in which two technologies can communicate data, allowing the relational database structure to be converted into an object-oriented format PHP can then use and manipulate [3].

Doctrine is the Symfony package bundle required to achieve a successful ORM with the MySQL database. The first step is to create Symfony entity classes for each table in the database. Within each class, there are variables mapping to each column in its related table. Here, constraints can be placed within code documentation blocks above each field variable. This may include field type, character lengths, whether the field is nullable, and more. Getters and setters are also created for each column in the table. The entity can then be used in conjunction with an entity manager to programmatically create, edit, and delete any row in a target table [22].

Using Doctrine and ORM provides an extra layer of security for the database. Direct queries against the database can be confusing and dangerous, leading to irreversible

changes if an actioned query causes undesirable behavior. With Doctrine, the entities will be validated before they are placed in the database. PHP will return a verbose error if a request is invalid. Finally, using Doctrine minimizes development time by providing easily understandable methods and functionality [22].

Figure 14: Roots User Entity Class, PHP. Source: https://github.com/hayleemillar/thesis-project



```
UserRepository.php
src > Repository > UserRepository.php > ...
You, 7 months ago | 1 author (You)
<?php

namespace App\Repository;

use App\Entity\User;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;
use Symfony\Component\Security\Core\Exception\UnsupportedUserException;
use Symfony\Component\Security\Core\User\PasswordUpgraderInterface;
use Symfony\Component\Security\Core\User\UserInterface;

You, 7 months ago | 1 author (You)
/**
 * @method User|null find($id, $lockMode = null, $lockVersion = null)
 * @method User|null findOneBy(array $criteria, array $orderBy = null)
 * @method User[]    findAll()
 * @method User[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class UserRepository extends ServiceEntityRepository implements PasswordUpgraderInterface
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, User::class);
    }

    /**
     * Used to upgrade (rehash) the user's password automatically over time.
     */
    public function upgradePassword(UserInterface $user, string $newEncodedPassword): void
    {
        if (!$user instanceof User) {
            throw new UnsupportedUserException(sprintf('Instances of "%s" are not supported.', \get_class($user)));
        }

        $user->setPassword($newEncodedPassword);
        $this->_em->persist($user);
        $this->_em->flush();
    }
```

Figure 15: Roots User Repository Class, PHP. Source: https://github.com/hayleemillar/thesis-project

In addition to each entity, repository classes are also required. When querying the database within code, an instantiation of the repository class allows the code to make easy queries and return arrays of results in the correct entity format, thus finalizing the mapping of the database into a usable PHP object [22].

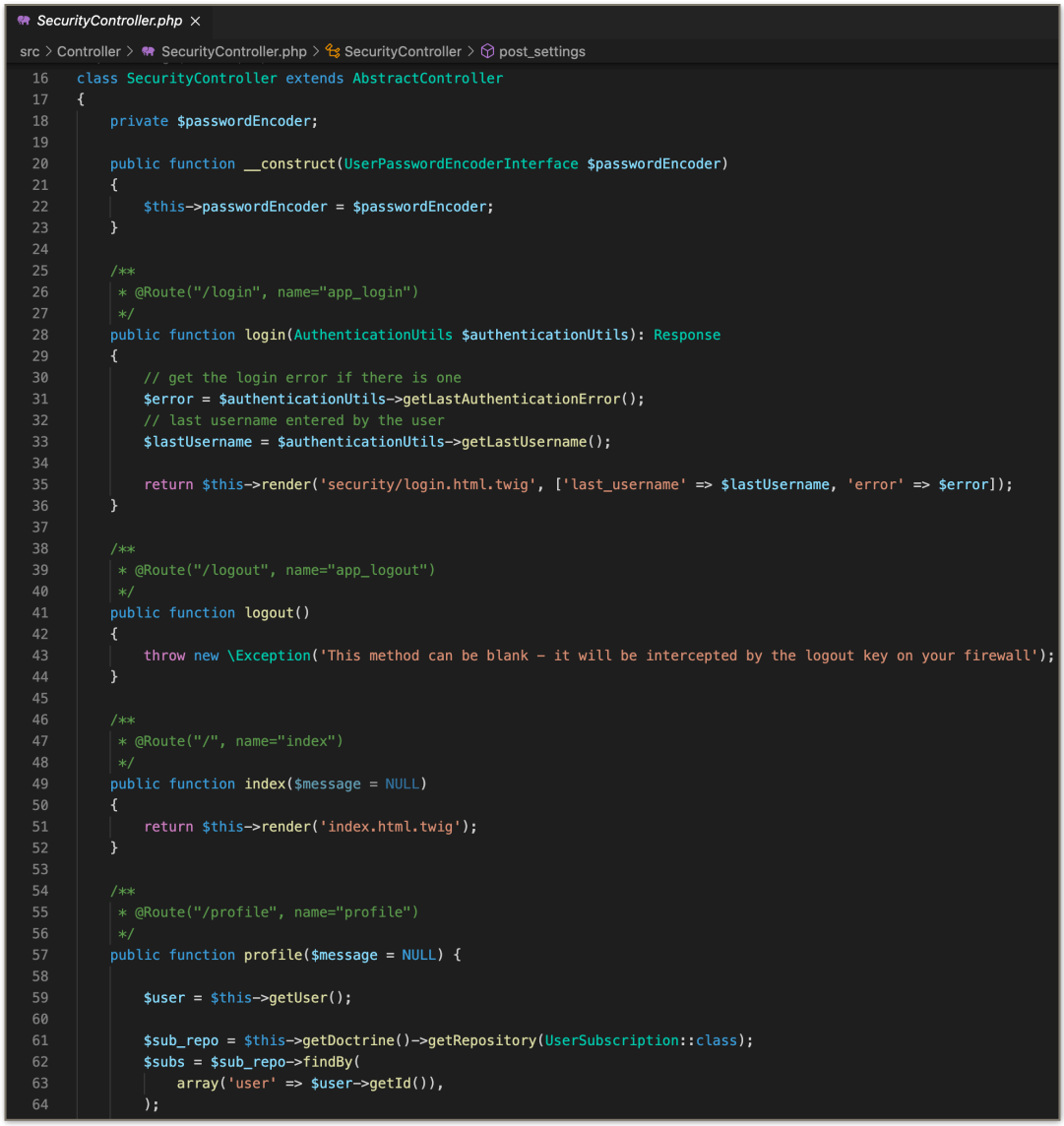


Figure 16: Roots User Security Controller. Source: https://github.com/hayleemillar/thesis-project

Controllers are how Symfony manages URLs, redirects, and HTML template rendering. A controller is a PHP function within a class where each method manages a page path. Behavior, calculations, and function calls occur within the class to return the proper information to the user. The controller can connect to the database by using Doctrine and entity classes. Using the entity manager and entity repository object instantiations allows

developers to manipulate data. Controllers are where users can be validated in the backend service, which is an important security implementation. If a malicious third party is attempting to send bad data or manipulate the database, validating the user creates a check to prevent data tampering. Users attempting to access a page they are locked out from will result in a redirect to an unauthorized notice page [20].

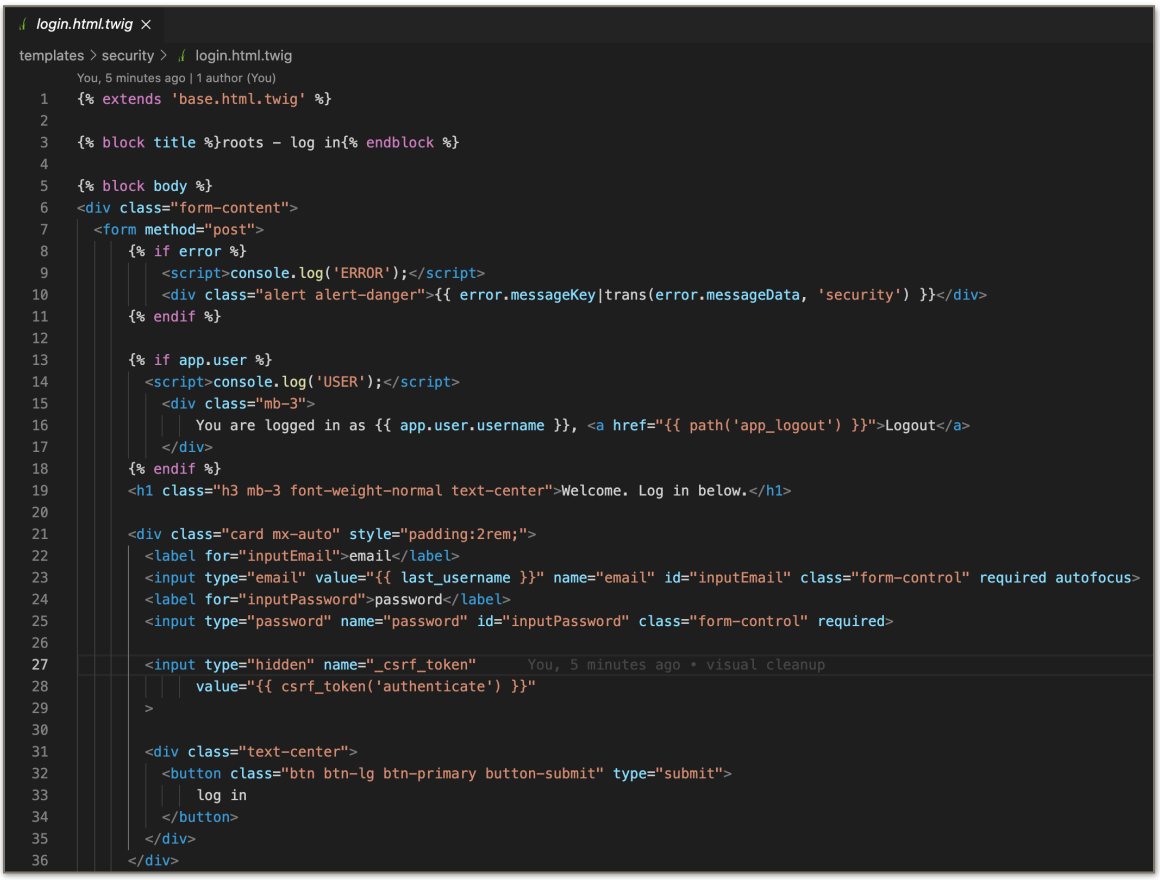


Figure 17: Roots Login HTML Template. Source: https://github.com/hayleemillar/thesis-project

The HTML loaded by the controllers for each given route is a rendered Twig template. Twig is a templating language that simplifies HTML, making development easier and quicker. The syntax is easy to understand if a developer is already familiar with HTML

and basic programming principles. An advantage to Twig is its accessibility to PHP variables which are passed to the template from the controller. It can also compile different HTML chunks based on logic expressions [21].

Once a user has subscribed to a tier for a given page, the behavior must be in place to securely validate the user from Roots to the external content website. Data must be sent to inform the website of the user's valid subscription. The website must be able to receive and make behavioral decisions based on the data received. It must be decided what content, if any, to show the inbound user.

JavaScript Object Notation (JSON) is a common data format used on the Internet when exchanging data. It is easy for both humans and computers to read, making it powerful and accessible to most programmers. It is readable by several programming languages. A JSON object is composed of key-value pairs.

JSON Web Tokens (JWTs) are digitally signed tokens that deliver encoded data. JWTs will allow for the authentication of inbound Roots users to content websites. Inside the encoded data will be information about the user and their subscription tier level. The tokens must be digitally signed because it ensures they are valid and have not been tampered with by a third party [1].

When a user wants to leave Roots to consume content on a creator's website, the website will need to receive the JWT. The token is sent via the query string, which is a parameter value within the URL (e.g., http://example.com/?parameter=SomeValue). Roots provides

a baseline Javascript file to place on the receiving content site, which decodes, validates, and caches the token.

Figure 18: Roots JWT Demo displaying first the encoded token followed by the decoded value.

The maintainer of the website then needs to integrate a provided Javascript file so that it is executed on every page. The file will check for an incoming token via the query string and place it within the website's cache. After the token is integrated into the caching system, the maintainer is required to create website behavior based on the token received. The token contains the user's subscription tier, so it will be important to alter the user's level of access to content based on the cached token. The implementation of the behavior requires programming and website development knowledge.

Chapter 5: Limitations and Future Work

There are a few limitations regarding Roots. First and foremost, the payment system needs to be applied in an official capacity, as the platform does not currently handle or process any transactional data. Official payment processing was determined out of scope for the first iteration of the platform. Stripe is a very popular tool for payment processing (https://stripe.com/), which takes the responsibility off of Roots as an entity to make a secure payment system. Stripe would be responsible for the payments and ensuring they are properly enacted.

Next, JWT requires site maintainers to have developer knowledge to apply the proper token behavior. Not only are they required to know where to put the provided Javascript file, but the developer will also need to know how to handle cached variables and make informed decisions in the backend of their website to ensure the dataset in the JWT is being handled properly. Such a knowledge threshold is high and undesirable, as the goal is to make Roots accessible to as many creators as possible. Future work includes creating modules/plugins for various content management systems (CMS) like WordPress to enable those who do not code to be able to customize websites as well. The modules will allow users to configure the JWT handling via a graphical user interface (the CMS) rather than coding a custom handler. It is common for developers to use open source tools to program and release custom modules for others to download and use. Since Roots is built on FOSS, modules for popular frameworks are believed to be achievable in a future iteration. Until then, users will be required to have programmatic access to their websites.

Finally, the use of a query string to send the JWT is not ideal as it makes the URL very long, which could be confusing to users unfamiliar with URL structure. Future work will include exploring better ways to deliver the token. One such solution could be creating a receiving route in the module solution mentioned above to receive a POST request with the dataset as the payload. It is easily achievable and can be included in the same iteration as creating modules for popular CMS frameworks.

Conclusion

Roots is a unique subscription platform not yet existent online for digital media. It has a vision of what other similar-minded companies lack; both a platform where creators can be found and a solution that allows those creators' autonomy over content presentation. Patreon has a platform but users are restricted to placing their content on Patreon's pre-styled pages. The creator cannot change the styling of their posts page. Memberful lacks a central platform for users to explore and find new creators. Podia requires users to host their website with Podia, as well as lacking a central platform like Memberful.

Both the creator and consumer experience are aimed to be bolstered by the design of Roots. The homepage attempts to draw users in with explanations, reviews, and calls to action. The explore pages allow users to either explore trending creators or search specifically. Content creators may own landing pages where they can advertise their websites. Tiers are attached to the landing page, allowing the creator to customize their monetization to their liking. Finally, all users can view their subscriptions and landing pages under their profile page in simple table formats to provide abbreviated information.

Open-source software is the pinnacle of the system architecture for Roots. Linux, Nginx, MySQL, and PHP (LEMP) comprises Roots. Their large communities online provide a wealth of resources to consider when troubleshooting an issue. There are thousands of maintainers around the world ensuring the security and progression of the software. LEMP provides a base to Roots on which Symfony interacts. Symfony handles the HTML generation for routes and database interfacing. Using compatible PHP packages like Doctrine, Symfony executes server-side functionality that makes changes to the database and provides users with information.

Using JWT for authentication, users can easily leave Roots and enter the content website they want to visit. The token is stored as a query string within the URL; website maintainers are provided a Javascript file that should execute on every page of the website checking for the token. If a token is received, it is cached, where the maintainer is then required to add programmatic token checking behavior based on information cached.

Future work first requires a payment system to be applied so the goal of monetization can be fully achieved. Following the payment system would be expanding Roots to be accessible to a wider creator audience. Content management systems like WordPress will be considered for plugin publishing, where non-programmatic users will be able to download the plugin and use it to monetize their website with Roots. Finally, the token is desired to stop being passed as a query string in the URL, which can also be achieved through plugin creation.

Glossary

Ad Blocker: An Internet browser plugin allowing users to remove all advertisements from web pages.

Apache: A popular web serving software that handles incoming requests for web page access.

Cache: Temporary data storage for website data within the user's device. It is commonly employed to help the user load their most frequented websites quickly.

Call to Action (CTA): Phrases used to engage the user into doing a specific action.

Closed Source Software: Proprietary software created and licensed by a person or company. Those in possession of such software may license it for monetary gain.

Content Management System (CMS): Software that provides a graphical user interface for non-programmatic users to modify a website's content. Examples include WordPress and Drupal.

Controller (Symfony5): For the Symfony framework, a function within a class used to handles a request, do server-side computations, and produce a response.

Database: See *Relational Database*.

Db2: Proprietary software developed by IBM with relational database functionality in mind. IBM has developed Db2 to include non-relational data as well.

Django: A Python framework designed for building web applications. Its strengths are emphasized in less high-level code needed to achieve higher functionality than traditional services like PHP.

Doctrine (Symfony): A series of PHP dedicated to object-relational mapping for databases.

Free & Open Source Software (FOSS): Software that is both free and its code is publicly available to be maintained by a community of developers.

GitHub: A website often used as a version control system for code.

HTML: *Hypertext Markup Language*; A markup language saved as text files to create web pages.

HTTP Daemon: Also known as *HTTPd;* A program on a server that handles incoming requests for a website. It will return requests and serve files.

JavaScript Object Notation (JSON): A common format used in web applications to send data.

JSON Web Token: Encrypted data with a digital signature in JSON format, which can then be sent across the web and verified by a receiving party of its validity.

LEMP Stack: Linux, Nginx, MySQL, PHP; A group of FOSS software commonly used in industry for web applications.

Linus's Law: A claim for software development that "given enough eyeballs, all bugs are shallow".

Linux: An open source operating system.

Load Balancing: Distributing incoming requests across services to reduce user load times.

Memberful: Software that allows websites to monetize their content, which allows users to integrate the subscription model directly into their website.

MySQL: Open source software that allows for the creation and manipulation of relational databases.

New York Times: A popular newspaper company.

Nginx: Open source software that handles incoming requests for web applications with other practical applications as well like load balancing and caching.

Object Relational Mapping (ORM): A practice in computer science employed to map data between incompatible software types.

Parameter: A value passed to a function in code to be used to determine behavior or make calculations. A value could be text, numbers, or some data type format readable by the software.

Patreon: A platform where users can subscribe to content creators. Each creator has a web page where users can view their content.

PHP: Short for *Hypertext Preprocessor*, a scripting language commonly used across the Internet for web applications.

Podia: A platform where creators host their websites. Through the provided tools they can monetize their platform through subscriptions.

PostgreSQL: Open source relational database software.

Python: Popular programming language for web applications among other applications.

Query: A request for data from a database.

Query String: A component of a URL that contains parameterized data, which can be accessed programmatically.

Relational Database: A type of database with structured data, which has relationships that can be leveraged within queries.

Scalability: The ability for a server to handle a change in user traffic that either increases or decreases resource demands.

Stripe: A payment processing platform.

Structured Query Language (SQL): A language specific to constructing queries to a relational database management system.

Symfony5 (Symfony): A PHP framework specialized for building web applications in a modular format.

Technology Stack: The languages and software used to build an application.

Total Cost of Ownership (TCO): The total requirement in money and resources required to manage goods or services.

Twig Template: Templating software for PHP for formats like HTML, CSV, and XML.

Ubuntu: A popular Linux OS distribution.

Uniform Resource Locator (URL): An address for the Internet where one can find data or an application.

Extensible Markup Language (XML): A markup language specialized to be readable by both humans and computers. It provides a set of guidelines for data to adhere to.

Works Cited

[1] Auth0. JSON Web Tokens Introduction. Retrieved from https://jwt.io/introduction.

[2] Ben Balter. 2015. 6 motivations for consuming or publishing open source software. (December 2015). Retrieved from https://opensource.com/life/15/12/why-open-source.

[3] Christopher Ireland, David Bowers, Michael Newton, and Kevin Waugh. 2009. Understanding Object-Relational Mapping: A Framework Based Approach. International Journal on Advances in Software 2, 2 & 3 (2009), 202–216. DOI:http://dx.doi.org/10.1.1.361.9922

[4] Daniel Oh. 2018. What is a Linux server and why does your business need one? (May 2018). Retrieved from https://opensource.com/article/18/5/what-linux-server#:~:text=These%20are%20designed%20to%20handle,stability%2C%20security%2C%20and%20flexibility.

[5] Django. Homepage. Retrieved from https://www.djangoproject.com/.

[6] Gary Henderson. 2020. What Is A Content Creator? (May 2020). Retrieved from https://www.digitalmarketing.org/blog/what-is-a-content-creator.

[7] Girish Punj. 2013. The relationship between consumer characteristics and willingness to pay for general online content: Implications for content providers considering

subscription-based business models. Marketing Letters 26, 2 (2013), 175–186. DOI:http://dx.doi.org/10.1007/s11002-013-9273-y

[8] IBM. Relational Databases. Retrieved from https://www.ibm.com/cloud/learn/relational-databases#:~:text=The%20primary%20benefit%20of%20the,group%2C%20and%20also%20combine%20queries.

[9] Maryville Online. 2020. What Is Digital Media? All You Need to Know About New Media. (March 2020). Retrieved from https://online.maryville.edu/blog/what-is-digital-media/.

[10] Memberful. Pricing. Retrieved from https://memberful.com/pricing/.

[11] The New York Times. 2018. Homepage. (August 2018). Retrieved from https://www.nytimes.com/.

[12] The New York Times. The New York Times Twitter Account. Retrieved from https://twitter.com/nytimes?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor.

[13] The New York Times. The New York Times Instagram Account. Retrieved from https://www.instagram.com/nytimes/?hl=en.

[14] NGINX. 2018. What is NGINX? How different is it from Apache (for example)? (September 2018). Retrieved from https://www.nginx.com/faq/what-is-nginx-how-different-is-it-from-e-g-apache/.

[15] Opensource.com. What is open source? Retrieved from https://opensource.com/resources/what-open-source.

[16] Patreon. Retrieved from https://www.patreon.com/product/pricing.

[17] Podia. Pricing. Retrieved from https://www.podia.com/pricing

[18] Southern Tide Media. 16 of the Best Website Homepage Design Examples. Retrieved from https://www.southerntidemedia.com/16-of-the-best-website-homepage-design-examples/.

[19] Stack Overflow. Where Developers Learn, Share, & Build Careers. Retrieved from https://stackoverflow.com/.

[20] Symfony. Controller (Symfony Docs). Retrieved from https://symfony.com/doc/current/controller.html.

[21] Symfony. Creating and Using Templates (Symfony Docs). Retrieved from https://symfony.com/doc/current/templates.html#twig-templating-language.

[22] Symfony. Databases and the Doctrine ORM (Symfony Docs). Retrieved from https://symfony.com/doc/current/doctrine.html.

[23] Symfony. Symfony at a Glance. Retrieved from https://symfony.com/at-a-glance.

[24] The PHP Group. What can PHP do? Retrieved April 17, 2021 from https://www.php.net/manual/en/intro-whatcando.php

[25] Yuping Liu-Thompkins. 2019. A Decade of Online Advertising Research: What We Learned and What We Need to Know. Journal of Advertising 48, 1 (2019), 1–13. DOI:http://dx.doi.org/10.1080/00913367.2018.1556138.