On Magnolia Square: Website Development Proposal

Neo Alabastro

October 7, 2023

Summary

How to Read This Document

I recommend a thorough read from start to finish, especially for those who are not as technically competent. Those who are technically savvy may skip §2.1. For those who are only financially curious about exact proposed costs, §4 provides financial information under each subsection. I sincerely thank you for reading my proposal, whether it be digested through thorough ganders or cursory glances.

1 Introduction

This proposal, geared toward those without a technical background, presents services, tools, and strategies to aid On Magnolia Square's members who are interfacing with the website's aesthetic architecture, content management, and technical stability, considering the lowest common denominator of technical literacy. Each proposed service or tool consists of general information and financial costs. This proposal also describes team roles needed to complete a full transition from the current website to a new one, and concludes with a sequential phase roadmap. This introduction section provides purposes and explanations for the rhymes and reason for this proposal's creation.

1.1 Problem

For the On Magnolia Square (OMS) team, the current OMS website has an obtuse and unwieldy website management services and tools required for its maintenance. For the reader, the OMS website lacks important security fixes and a cohesive user interface and experience, consequently reducing readership and public interest.

1.2 Goal

My goal is to develop and deploy a new OMS website that will not only repair the current website's issues of security, aesthetic, and content management, but one that will also serve as a stable foundation on which current and future OMS members, regardless of their technical ability, may be able to create, maintain, update, and manage self-published articles, photos, and content.

1.3 Solution

The solution is to employ modern services and tools to achieve the aforementioned goal, while also providing proper documentation for each part of the process for future OMS members to reference.

1.4 Purpose

Throughout this organization's history, no members have indulged in an in-depth proposition to change the OMS website architecture. This organization's leadership frequently changes hands, so many iterations to the website's identity have manifested organically creating what it is today. It is functional, but given the problems in §1.1, its potential to be a cornerstone of the New York University Shanghai (NYUSH) community is unnecessarily limited, even with the many shades of undergraduate talent intrinsic to our academic atmosphere. Completely online publications like OMS are generally very flexible in their execution, so I believe it is important to reify a logical game plan present and future members can utilize to focus on guaranteeing purposeful journalism.

2 Terminology

For the purpose of speaking the same language, this section introduces terminology required to understand the rest of this proposal.

2.1 Technical Terms

Understanding the construction of a new website requires familiarity with website terminology. Please remember that this reading is geared toward those with non-technical backgrounds, and does not require intimate knowledge of web technologies to understand.

The Internet is analogous to a highway: a series of roads that connect distinct locations, like your computer or mobile device, to other distinct locations, like websites and remote servers. Traveling on the highway is typically done using a web browser. Similar to a personal chauffeur, a web browser prompts you for an address. Where do you want to go? That address is called the **Uniform Resource Locator (URL)**. These addresses are composed of several parts, but what matters for our purposes is the **domain name**. For instance, take this URL:

https://www.google.com/search?q=nyu+shanghai

The domain name is *google.com.* Other examples of domain names are punahou.edu, minecraft.net, or onmagnoliasquare.com. Another term that often comes up is with the term domain name is **Top Level Domain (TLD)**, which just means the two to three letter segment after the dot. For example, edu, net, and com are all TLDs. For our domain,

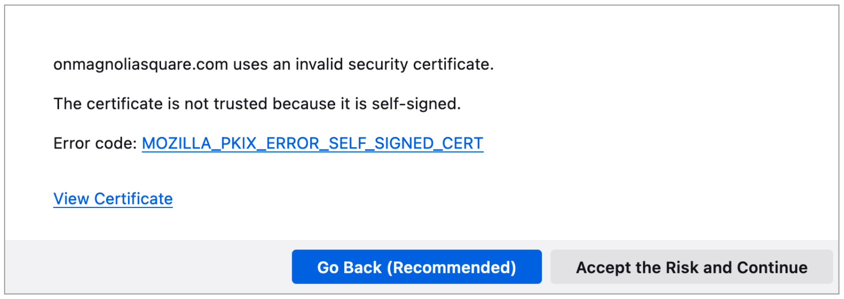
https://www.onmagnoliasquare.com

The domain name is *onmagnoliasquare.com.* The TLD is *com.*

A **domain name registrar** provides domain name registration and leasing services. There are many domain name registrars, like GoDaddy, Namecheap, or Amazon Route 53. Domain names are leased on a yearly basis. Registrars often give the option to pay for two, five, or even 10 years in advance. Our current domain name registrar for OMS is GoDaddy.

Sometimes, domain name registrars offer **website hosting** services. For example, if a website was a building, some domain name registrars do not only let you register an address for your website, but they also lease a plot of land you can build your building on; they lease a physical space people can actually see and visit.

Now, imagine that the web browser is your personal chauffeur once again. They take your provided URL address, https://onmagnoliasquare.com, and drive you to the OMS website. Before unlocking the doors and letting you off at the location, they check in with the security at the gates. Your chauffeur wants to validate that the address you provided and your current location is indeed the place that you want to be. In other words, you gave your chauffeur the address for the OMS website and now that you have arrived, they want to ensure that yes, this location is actually the OMS website, or no, this is not the OMS website, and the address you provided is wrongly associated with the current location. This validation is done with a **certificate**[1], a cryptographic electronic verification validating true ownership of a domain issued by a type of organization called a **Certificate Authority (CA)**. Your chauffeur rolls down the window so location security can hand over their certificate for your chauffeur to inspect. Upon inspection, your chauffeur will either let you off or freak out. In the case that they freak out, it is most likely for this reason: the certificate from location security is a fraud! It is self-signed, meaning it was not issued by a valid CA. They turn to you and say: 'Hey, there's an issue: your personal security may be at risk if I let you off at this location. After inspecting this location's certificate, it appears to be self-signed. Do you still want me to drop you off here?'


onmagnoliasquare.com uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

Figure 1: A self-signed certificate warning in the Firefox web browser.

The current OMS website suffers from having a self-signed certificate. It has been a persistent issue for over a year to this date. It drives readers away even though our website is not harmful.

All the terms I have described thus far refer to the **backend**. It is infrastructure that our typical website reader is unaware of, nor needs to know anything about. Maintaining an online presence using a website is like hosting a theatrical play. You want an audience to come and watch. They do not care about the stagehands or the prop designers or even the words on the script or anything else happening off-stage. They care about the performance and the plot. They care about the **frontend**.

The frontend is typically associated with the term **User Interface and User Experience (UI/UX)**. It simply describes the qualities of the way a user of something interfaces with it, and the experience that accompanies using that interface.

For example, a wooden pencil has a simple user interface: a slightly long, cylindrical instrument beginning at a short rubbery stub, and terminating at a fine, cone-shaped tip. A pencil also has a non-problematic user experience: users

[1]In technical terms, this is called an SSL X.509 Domain Validated Certificate.

can hold it in any which way they please without impairment to handwriting or drawing technique. These two characteristics of its UI/UX grant the user the ability to apply the graphite tip to a flat and markable medium in a sufficient and comfortable manner, such that any hand movements required to produce a writing or drawing can be done effectively, efficiently, and easily.

The content of many journals or newspaper websites, for example, the articles you read on JSTOR or The New York Times, is created, updated, and deleted using a **content management system (CMS)**. A CMS interfaces between the frontend and the backend. It takes information from the backend, like articles or images, and sends them to be displayed at the frontend. A user who wants to update content on a website uses a CMS. A CMS is a tool developed for publishers, editors, and content creators, and so the UI/UX of a CMS is accessible for those genres of work.

A CMS can be imagined as the pencil in the UI/UX example. It interfaces between your ideas and the physical or digital medium on which the idea will be presented. A CMS with a good UI/UX should feel fluent and fast, similar to how a pencil with a good UI/UX should feel comfortable and familiar.

2.2 Qualitative Terms

Each proposed system or backend and frontend service is chosen for its demonstration of good qualities in three categories of traits:

1. Accessible
2. Robust
3. Future-proof

Because of these traits, these services or tools can be used in the hands of a novice or experienced OMS member, with varying levels of technical ability, to achieve a satisfying outcome regardless of their goal.

Accessible

Accessible backend services boast a user friendly UI/UX. The OMS members operating the website in the future must have a successful and satisfying experience. Malfunction should not result from bad design. Accessibility also means that technical literature on the service is easily accessible from online documentation, online forms, or tech support.

Robust

A robust service is one that must be able to be kicked and tossed to the ground without compromise to its dependability. It is also mistake agnostic – an ignorant mistake or a well conceived blunder by an OMS member using the service will not hinder the service's reliability to display the website to our audience. In the case where an issue occurs, information to fix it is readily accessible.

Robust services also beget high standards of security, in the form of accessible user access management systems, multi-factor authentication, or reliable customer support.

Future-proof

Regardless of a service's accessibility and robustness, if it will not exist in the future, it is not employable in the present. Future members members must be able to access and use the service. I have chosen services created by strong and reputable companies in their respective technical circles, such that if a breaking change for their customers were to ever emerge down the line, a warning or notice will have been announced several months beforehand. This gives time for members to explore other service solutions.

3 Infrastructure

This section will discuss the current backend and frontend services and how they are put together. Each section will have a diagram depicting the system that creates the OMS website, comprised of the services involved and their interactions with each other. Arrows in each section depict the flow of content and information.

3.1 Current Infrastructure

Our current infrastructure uses GoDaddy as an End-To-End (E2E) solution. An E2E system has a solution for every part of the process of a deployment pipeline. For us, GoDaddy accommodates the entire website publishing spectrum, beginning from the writing and drafting, to publishing and design, and concluding at the eyes of the audience. Every service is consolidated under GoDaddy making it a one-stop-shop for everything, begetting convenience for simple content management.

I suppose E2E infrastructure was implemented because the GoDaddy route at the inception of OMS (previously On Century Avenue), was a conventionally tried-and-true, albeit severely sub-par, method of website creation.

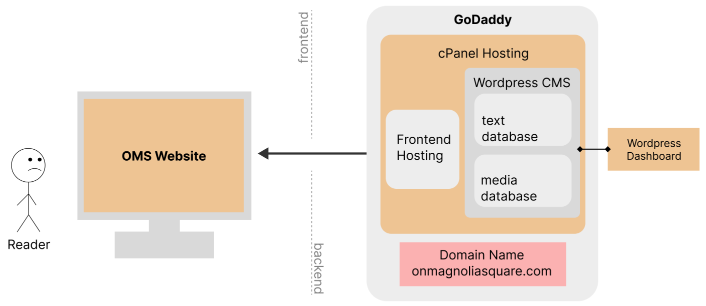


Figure 2: An End-To-End (E2E) infrastructure plan.

GoDaddy provides website hosting under *cPanel*. cPanel hosts our Wordpress CMS, its accompanying text and media database, as well as the frontend. Content flows from the Wordpress dashboard, the place where publishers input text and images, into a text and media database, and then onto the frontend where readers can view the content.

3.1.1 Issues

Although all services are consolidated under one roof, this approach is prone to vendor lock-in and rigidity. In terms of §2.2, this system is somewhat accessible: the system operates on less surface area, which can be good since all services are in one place. However, it violates §2.2 Robustness and Future-proofing, because where it lacks in surface area it makes up in density. Density is how internally complicated and convoluted it can be to interface with the system in order to change, update, or improve it. I have personally explored the underbelly of GoDaddy's hosting dashboard and it is not pretty: the UI/UX can be confusing, sluggish, unintuitive, and prone to user error.

3.2 Proposed Infrastructure

My proposed infrastructure uses a Best-Of-Breed (BOB) solution. A BOB system is like composing a three-piece suit from materials sourced from the best silk and cotton mills; every component that constitutes the website is the crème de la crème, or for our purposes, the ones that suit our requirements the best.

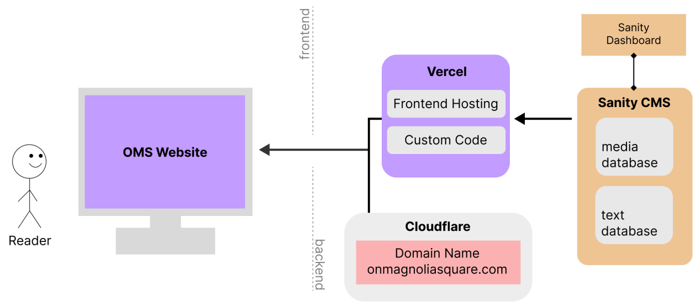


Figure 3: A Best-Of-Breed (BOB) infrastructure plan.

The flow of content follows a similar path to the GoDaddy E2E solution. It starts at a CMS dashboard, through to the CMS host, then to the frontend host, which is viewed by a reader when they access our domain name. The difference is the use of multiple services: Sanity is used as the CMS, Vercel is used as the frontend host, and Cloudflare is used as the domain name registrar.

Although BOB systems requires more surface area, the density of each service is manageable. For example, BOB systems are easier to debug and diagnose, because the components of the system are partitioned into discrete units with single responsibilities rather than consolidated into one box.

3.2.1 Issues

It requires some technical knowledge to know how these services interact with each other; some assembly is required. This means that there must be at least one OMS member technically oriented and also motivated to diagnose and fix an issue if it arises in the future. This may appear to violate §2.2 Future-proofing, because whoever joins the team may or may not be technically oriented, but the only hurdle to overcome this issue is knowledge and understanding. Anyone can execute it.

3.3 Ditch or Fix?

Distributing information online is always a sprinter's marathon: to achieve long term success, you must always be the quickest to adopt new tech. Therefore, rather than fixing the current infrastructure and ensuring that it maintains longevity into the future, I feel that it is better to ditch it, because the time spent attempting to fix and then ensuring the qualities of §2.2 are satiated is better spent on creating new infrastructure from the bottom up. Our small publication will require rocket engines strapped to its back to stay relevant. That cannot be done with our current infrastructure.

4 Proposed Services and Tools

This section proposes new and better backend and frontend services that will enable the smooth operation of maintaining the website's infrastructure, development, and content. Each proposed service also includes the issue that they fix. Prices are in United States Dollars (USD), as these are the prices provided on each service's website.

4.1 Sanity

Sanity is a *headless* CMS. Our current CMS, Wordpress, is not a headless CMS, because it also serves a frontend that shows the content from the CMS to readers (§3.1 fig. 2). A headless CMS gives more flexibility to designers and web developers who want granular control over the UI/UX of the frontend product; the way content is displayed and the content itself are decoupled entities.

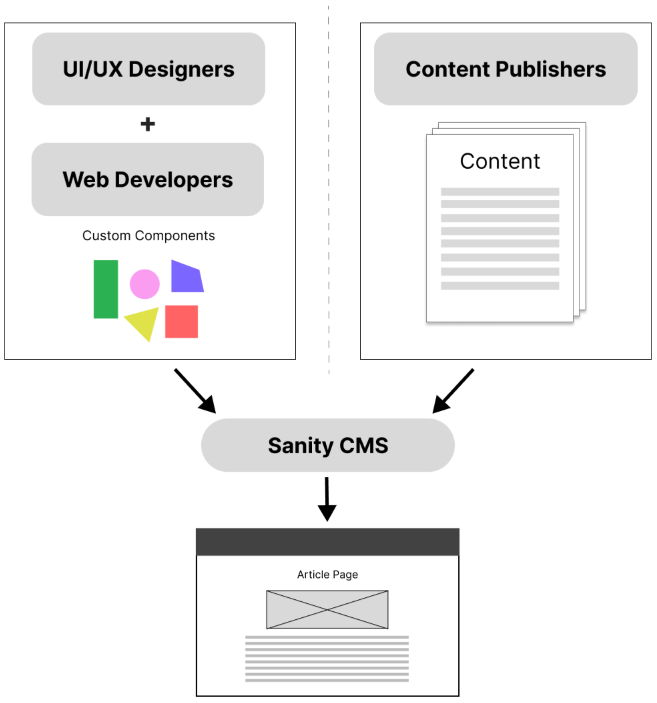


Figure 4: Sanity combines components and content for the frontend.

Features

Since content and UI/UX design are separated, writers and publishers will not need to treat design as an afterthought. Sanity translates the results of the collaboration between web developers and UI/UX designers into a domain in which non-technical publishers are able to design a page, regardless of the page's content, to their liking. Designers create the design of UI/UX components, which are essentially pieces that come together to form the look and feel of the web page. They pass the design schemes to web developers to implement them into real code. Sanity consumes this component code and prepares it for writers and publishers to incorporate into their content on the frontend. Publishers can then combine both written content and custom components to create articles or web pages that are published on the frontend, without having to design or write a line of code. This approach can satisfy the customizability needs of the website to have a more coherent UI/UX design language, as well as future-proofing since components created now can be used well into the future.

Price

Sanity's pricing is transparent[2]. Many platforms gate functionality by price structure – higher priced tiers receive better functionality. Thankfully, nearly everything we need is provided in their first tier, the 'Free Forever' plan.

Their pricing structure more-so delineates available usage quotas of different resources they provide in every tier. Similar to a phone plan, these quota limits describe how much data we are using per month to operate on their platform. I am most concerned with these five resources, in order of priority:

Assets	\$1 per 2 Gigabytes additional Assets per month
CDN Requests	\$1 per 250,000 additional CDN Requests per month
Documents	\$2.5 per 1,000 additional Documents per month
Bandwidth	\$1 per 5 Gigabytes additional Bandwidth per month
API Requests	\$1 per 25,000 additional API Requests per month

An asset is any type of digital media. Assets on our website are images or audio. CDN Requests, Bandwidth, and API Requests are all related to frontend to backend communication, or values that quantify how much communication is happening. Documents is the number of data stored in the text database[3].

Below is a small table categorizing the quotas into two categories: Fluctuating and Fixed. The Fluctuating category is based on communication usage, like texting and call data on a phone plan: the more you text and call, the more money leaves your wallet. The Fixed category is based on storage usage, like Google Drive. It is called 'Fixed' because it does not fluctuate as drastically as communication usage. The rate at which storage usage changes over time is significantly longer than communication usage based on a shorter time interval like those in the fluctuating category.

Fluctuating	**Fixed**
CDN Requests	Assets
Bandwidth	Documents
API Requests	

The Free Forever plan offers a free 5 gigabytes of asset storage. Every byte after 5 gigabytes will then be a charge on our account. Using 6 gigabytes (5 gigabytes + 1 extra gigabyte) means we will be charged \$1 extra for asset storage monthly. Using 7 gigabytes (5 gigabytes + 2 extra gigabytes) means a charge of \$2 to our account. Over time, storage use usually increases. This means that

[2]https://www.sanity.io/pricing

[3]Refer to §3 for a refresher on the infrastructure.

using the new website will incur additional asset storage fees. Since our asset storage will most likely be used to store images rather than bulkier media like video, this storage usage increase will be slow[4].

The Free Forever plan also offers free storage for 10,000 documents. A document can be imagined as a literal paper sheet with information on it; newspaper articles, acceptance letters, advertisements, and identification cards all fall under the term ‘document’. We can store 10,000 of these. The small catch is that revisions to a document count as an additional document. Therefore, revisions on the platform should be kept to a minimum to ensure maximum space optimization. This storage increase will follow the same financial trajectory as the asset storage: the more we store, the more the permanent cost, but the time to reach these costs is extremely slow.

Unless we hit an interstellar home run and become famous overnight, the amount of CDN Requests, Bandwidth, and API Requests should not breach the Free Forever plan’s free quota thresholds. The only case when I suspect a breach would happen is when we are in the process of transitioning to the new website when we upload content to it in large batches, and even then, the fee charge will be minute given their prices of $1.

In theory, if we were to keep our margins within the given usage limits of Sanity’s Free Forever plan, we would spend absolutely no money. In practice, this is impossible, but the charges are meaningful, warranted, and reasonable given the size of our organization.

4.2 Vercel

Vercel provides website hosting, named *Vercel Frontend Cloud*, for frontend projects. Vercel manages the nitty-gritty of hosting the frontend of a website, and lets its users concentrate on the website’s design, development, and code, rather than being impeded by technicalities. Vercel is accessible because their documentation[5] for developers and management alike is easily found online. It is robust because of their global content delivery network[6]. Finally, it is future-proof because many high-profile companies, such as Adobe, eBay, and The Washington Post, implement their services[7].

Price

They have a very generous free tier plan[8]. At this moment, we do not need the Pro Plan, which is $20 a month, because the features are somewhat irrelevant

[4]On a more technical tangent, if images are uploaded in a lossy compression format like JPEG, and assuming that every picture is on average 5 megabytes, it would require 400 photos to use 2 gigabytes of storage. Even then, JPEG compression can reach sub-megabyte compression into kilobyte compression territory; if images were rather 600 kilobytes in size, it would require around 3300 images to use 2 gigabytes of storage. What a drastic decrease!

[5]https://vercel.com/docs

[6]https://vercel.com/features/infrastructure

[7]https://vercel.com/customers

[8]https://vercel.com/pricing

to our needs. If an upgrade is ever needed in the future, I will notify those who may be of concern. Therefore, the total price to implement Vercel into the website infrastructure is $0.

4.3 Cloudflare

Cloudflare services and maintains digital cloud infrastructure across the globe. Cloudflare has good qualities in the three trait categories. Cloudflare is accessible: the UI/UX is simple and clear. Access to different settings and functions is clearly labeled. In the event that something may be unclear, their online documentation[9] and blog[10] is accessible and digestible. The services they provide are robust, dependable, and future-proof because a sizable majority of the internet flows through their worldwide data centers. Simply put, cloud security and internet infrastructure is their domain. Because of this, they provide back-end service products we're particularly interested in: their registrar and image storage.

Domain Registrar

In the technical community, using GoDaddy is a sin; any other domain registrar is better than GoDaddy. I chose Cloudflare's product, named *Registrar*, because of its metric gathering and ease of use. Cloudflare is also one of several domain registrars that offer a free certificate[11] for any domain on their platform, along with enhanced customizability for certificate options and security[12]. Cloudflare's analytics and metrics aggregation is also top-tier.

Please note that readers who view the website will be unaware of this registrar switch – visiting the OMS URL will still display the same website content before the switch. It does not affect the frontend.

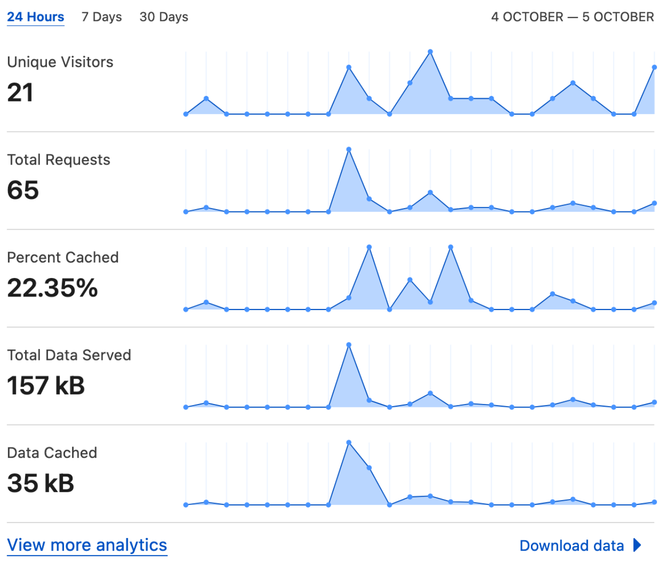
Image Storage

Cloudflare Images is Cloudflare's image storage solution[13]. Since the cPanel media database would no longer exist, Cloudflare is able to store and deliver any images we upload to Sanity. This service may or may not be used, depending on cost assessment of using Sanity's *Content Lake*. However, this option exists as a backup, and their prices are affordable.

Price

We are overpaying for our domain name at GoDaddy. They charge more than double that of Cloudflare's offering for our domain name.

[9]https://developers.cloudflare.com
[10]https://blog.cloudflare.com
[11]https://cloudflare.com/application-services/products/ssl/
[12]https://developers.cloudflare.com/ssl/get-started
[13]https://www.cloudflare.com/developer-platform/cloudflare-images/



Unique Visitors 21

Total Requests 65

Percent Cached 22.35%

Total Data Served 157 kB

Data Cached 35 kB

Figure 5: A view of the Cloudflare Registrar domain metrics dashboard.

At the time of this writing, Cloudflare's domain registrar sells domains with the *com* TLD at a flat rate of $9.77 a year[14]. Their image storage service starts at $5 a month to store 100,000 images. Delivering 100,000 images costs $1 extra for the monthly period. 'Serving' an image means displaying it on our website. If we were to serve 99,999 images, we would not pay $1. If we were to serve 500,000 images, we would have to pay $5.

The total cost *without* using Cloudflare Images is $9.77 a year. This does not include local taxes. The total cost with Cloudflare Images is $69.77 a year, since $5 a month equates to $60 for an entire year. This price does not take into account the fee for extra images served.

[14]A Cloudflare account is required to access the domain pricing list. Since I have one, I was able to retrieve this price.

5 Roles

I will build two teams: a design team and a tech team. The design team is responsible for creating the new UI/UX for the website. The tech team is responsible for code and technical utilities. The roles between each team is fluid. Those on the tech team can also work on the design team and vice versa. Members on each team will be selected and verified of their qualification for their prospective team by me, Neo Alabastro, and our current student director, Eva Weisenfeld.

I take the role of a delegate between three points of interest: technology, design, and accessibility. This project will not be the product of my own doing, but rather the collective effort of people with extensive skill sets in distinct disciplines. Therefore, my responsibility is to help them synthesize their work into a final product. In the case that I am unable to find certain people able to do certain types of specialized tasks, I will be responsible for that task's completion.

I will employ some guerrilla recruiting techniques such as word-of-mouth or spontaneous eavesdropping (I overhear something, I ask). I will also use more formal ones, such as asking for help emailing people of interest in the Interactive Media Arts major, as well as seeking website development help from peers in the newly established ACM student chapter at NYUSH.

6 Project Timeline

The entire project consists of four phases: Repair, Design, Develop, and Teach.

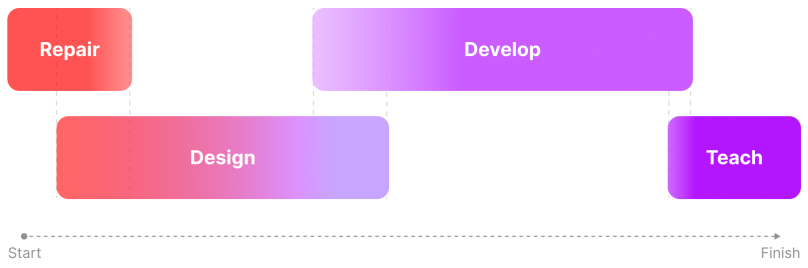
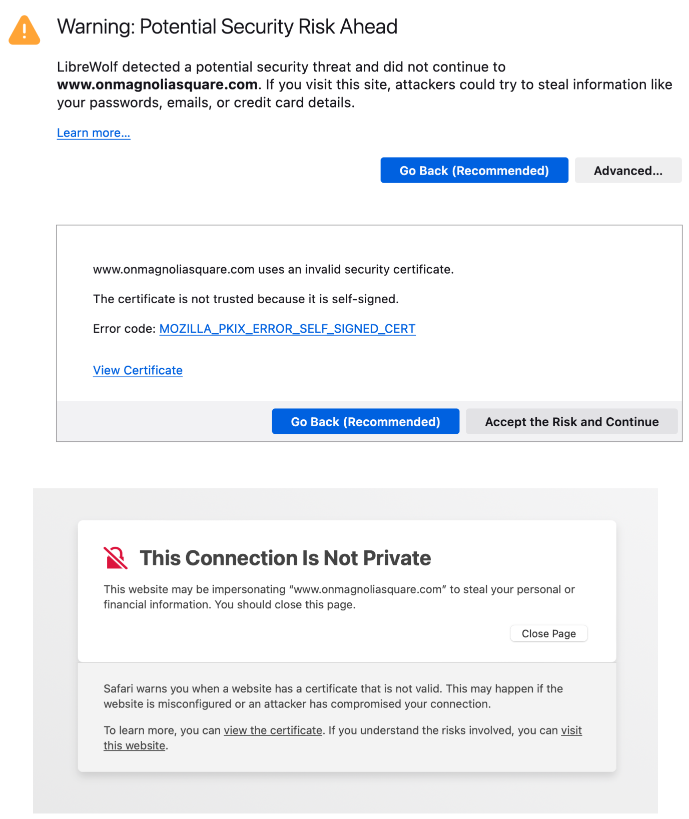


Figure 6: A simple graphic visualizing the phases.

The actual deadlines or sequential months are not given in figure 6. This should merely serve as a simplified timeline visualization. Note that these phases, although sequential, are also parallel. This shortens the time frame and design orientation, but requires more planning and organization.

6.1 Repair

This is a crucial phase that must happen as soon as possible. The goal of this phase is to fix the constant error that appears when visiting the site, as seen in figure 7.


Warning: Potential Security Risk Ahead

LibreWolf detected a potential security threat and did not continue to **www.onmagnoliasquare.com**. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

Learn more…

www.onmagnoliasquare.com uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

View Certificate

This Connection Is Not Private

This website may be impersonating "www.onmagnoliasquare.com" to steal your personal or financial information. You should close this page.

Safari warns you when a website has a certificate that is not valid. This may happen if the website is misconfigured or an attacker has compromised your connection.

To learn more, you can view the certificate. If you understand the risks involved, you can visit this website.

Figure 7: Flavors of the security warning, first in Firefox and next in Safari.

The issue is a website certificate error, a backend issue manifesting at the frontend. I do not yet know the reason for this error, but I am confident in my diagnosis. I have attempted to fix this error before, but my fix was unsuccessful, so moving to Cloudflare would eliminate all GoDaddy variables that may be causing the issue and expose more of the technical architecture for better surgery.

Moving to Cloudflare

Cloudflare has a step-by-step guide on their website detailing the steps and requirements to successfully switch domain registrars. In short, necessary credentials include:

- Credentials for GoDaddy
- Payment information to authorize domain payment
- Credentials for onmagnoliasquare@gmail.com

These items will enable me to successfully transition our DNS Registrar from GoDaddy to Cloudflare. Transferring a domain between registrars will not affect the domain name or TLD. Those will be exactly the same.

Caveat

Just because we change domain name registrars does not mean we are completely done with using GoDaddy in the meantime, since we still need GoDaddy to host the cPanel hosting for the Wordpress CMS. We leave GoDaddy at the end of the Develop phase.

6.2 Design

This phase takes place during the Fall Semester of 2023 and concludes into the middle of the Spring Semester in 2024. In this phase, we will make overhauls to the UI/UX design of the current website. It will feature a coherent experience from desktop to mobile devices, as well as compliance with web content accessibility guidelines. We will execute a UI/UX design development process, which starts with user research, to defining the problem, to actual designing, then to prototyping, then to feedback. Included in this process is the creation of usability interviews and studies, cross-platform coherent design system diagrams, and non-functional prototypes of the site itself.

6.3 Develop

During this phase, real world web development takes place, meaning a combination of asset and art creation, tech systems migration, and programming. The goal in this phase is to implement the new design system, to ensure technical stability across the platform for both readers and OMS members, and to start copying content from the old website to the new one. Once all content has been copied and technical stability is ensured, a transition to the new website will take place, and the old one wll be discarded. This phase will begin and end in the Spring Semester of 2024.

6.4 Teach

This final phase is about introducing and teaching the new website's features, functionality, and management. Teaching about these pivotal changes and functionality will occur during one or two team meetings. Due to the nature of a university student's social life and educational or non-educational commitments, I will personally reach out to those responsible for publishing content or maintaining the website in order to get them up to speed. Therefore, the meetings will be a general overview, and I will have personal meetings with respective OMS committee members for their benefit.

I will disseminate three reference documents. One with design documentation, one with development documentation, and one with general information.

Design documentation will explain the website's design language and the reasons for the choices. It will also provide a guide on reasoning through a UI/UX change. Development documentation will provide future OMS developers with help with the systems architecture and lessons we will have learned from developing the website. The general information guide will be for understanding the general functioning of the website in non-technical language. It is up to future members' discretion to implement this documentation I will have created. These are merely guides.

After the project is complete, I should not have to intervene with any issues after I graduate. However, such a high stake goal does not always manifest in the way you want it to. In the case that major issues arise that would perhaps require knowledged intervention, future OMS members are always welcome to email me in the future regarding how things work. I will be glad to teach them. Graduating from NYUSH only signifies my transition from a student to a mentor. I am always willing to give back to the community that created me.