Website Overhaul

Bryon Tjanaka (Webmaster) / Status: Draft 2020-10-24

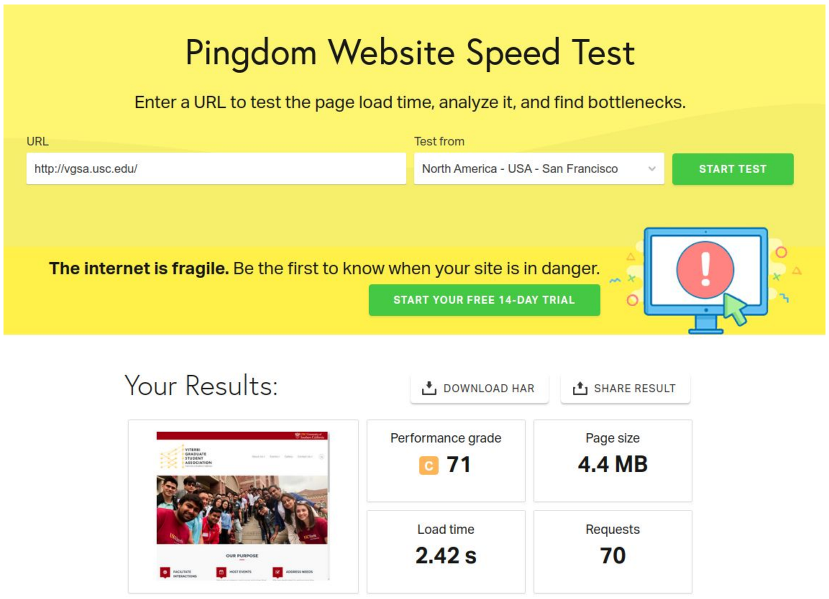
Objective

Redesign the VGSA website with modern technologies.

Background

The VGSA website is a Wordpress website hosted at http://vgsa.usc.edu. Though it currently serves its purpose, it suffers from several notable problems:

- **Poor performance:** According to Pingdom (link), it takes 2.42 seconds to load the home page, of which 1.4 seconds are spent waiting for the server to respond.


Pingdom Website Speed Test

Enter a URL to test the page load time, analyze it, and find bottlenecks.

URL: http://vgsa.usc.edu/

Test from: North America - USA - San Francisco

START TEST

The internet is fragile. Be the first to know when your site is in danger.

START YOUR FREE 14-DAY TRIAL

Your Results:

DOWNLOAD HAR

SHARE RESULT

Performance grade	Page size
C 71	4.4 MB
Load time	**Requests**
2.42 s	70

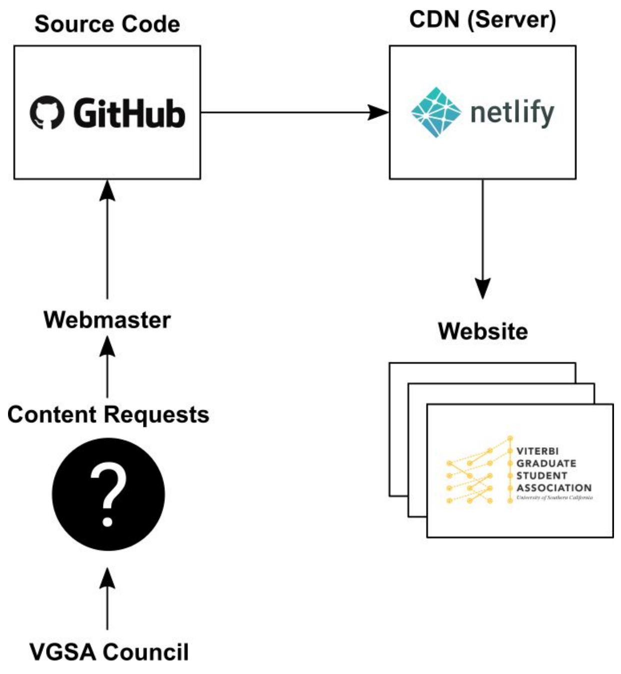
- **Lack of maintainability:** Since Wordpress is designed for users with little familiarity with code, the UI is optimized for ease of use rather than efficiency. Tasks like uploading team members are incredibly tedious to execute.
- **Lack of flexibility:** Many features which could easily be implemented from scratch are impossible to find in Wordpress, or they require a heavy plugin.
- **Lack of backups and version control:** A single copy of the entire VGSA website is stored on the GoDaddy servers, and there is no way to retrieve it. Backup software fails because the site is too large, and version control software like VersionPress fails because the server is not compatible. Essentially, if the GoDaddy server goes down tomorrow, the website would be gone.

Proposal

Given that the VGSA website mostly hosts static content, I propose to re-implement it using a static site generator that uses HTML, CSS, JavaScript, and other (programming) languages. The source code will be hosted on GitHub, and the website itself will be hosted on a CDN such as Netlify. This change will solve the above problems as follows:

- **Performance:** By converting the site to a set of static pages that are hosted on a CDN, our website is as fast as possible. We no longer need to worry about slow servers.
- **Maintainability:** The website will be written in several programming languages, which will make it easier to implement more complex features.
    - There is certainly a tradeoff here in terms of ease of use. Namely, future webmasters will need familiarity with JavaScript, but that should be easy to find. Furthermore, users other than the webmaster will have a harder time editing the website directly, but I believe we can solve this by setting up a process for other users to submit content to the webmaster.
- **Flexibility:** As we will be able to work in raw HTML/CSS/JavaScript, virtually anything is possible.
- **Backups and Version Control:** By using git and GitHub, we ensure we have backups of the site as well as version control.

Essentially, the website workflow will look like this when completed:



The VGSA Council and other clients will send the Webmaster (me) requests for website content. Then, I will implement the content on the website and push / upload the source code to GitHub. The CDN will automatically generate the website from GitHub, and it will serve the pages to end users.

Implementation

We seek to implement a highly customizable static site. In particular, we will need the following technologies:

- A static site generator, which generates the website from templates. Among other things, a static site generator makes it easy to reuse components across pages.
- A library that provides highly customizable web components, so that we do not have to design the entire website from scratch.

Static Site Generator

I will use Eleventy, a static site generator written in JavaScript. I did consider using Jekyll, a static site generator written in Ruby, as the authors have more experience in using it. While Jekyll certainly works, it is growing a bit old, and Ruby is also not an ideal framework. Choosing a JavaScript framework enables us to leverage the power of the JavaScript ecosystem.

Component Library

I will use the Bootstrap framework, which provides flexible components for webpages.
I considered using Materialize, a CSS framework based on Google's Material Design. Materialize provides components such as cards and navbars, and it is easy to integrate into a website. However, I decided to not use Materialize because:

1. The initial version of Materialize is now abandoned, and as such, I would be using the less popular new fork, with files hosted on jsDelivr.
2. Materialize is not as customizable as Bootstrap, as it is designed to conform with the Material Design guidelines.
3. Bootstrap has a much larger community and support system.

Finally, I did consider Material Components Web, but it seemed difficult to find documentation for integrating these components into static sites.

Interactive Components

I anticipate most of the website will be static content. However, in the case that I need to build highly interactive components, I will integrate Preact into Eleventy based on this tutorial. Furthermore, if I need to handle dynamic data, I will consider using a database like Firebase.

Prototype

Please refer to https://github.com/tjanaka/website for a small website I made that uses Eleventy, along with Webpack for handling assets. It can easily be extended with Bootstrap and Preact as described above.

Resources Requested

All of the aforementioned resources are either open-source or provided at no charge, so I am not currently requesting any funding.

Access to GoDaddy

In order to transition the website, I will need to be able to change DNS settings and phase out the old website. I am currently working with Juli to get access to the GoDaddy account, but I do not anticipate using it until this new website is almost ready for deployment. While in development, the website can be hosted on a separate URL.

Timeline

I do not have a set timeline on this overhaul. At latest, I will begin implementing it over winter break.