U.S. Cyber Range
Reinvigorating Cybersecurity Education

Final Report

CS4624: Multimedia, Hypertext, and Information Access
Instructor: Dr. Edward A. Fox
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA
December 8, 2021

Authors: Trenton Deel, Aiden England, Matthew Rogers, Vikramaditya Pratha

Publisher: Virginia Tech

Client: Jeff Joiner and the FourDesign Team

Table of Contents

Table of Figures

Executive Summary

The US Cyber Range is a cloud-hosted system for training students across the United States on a variety of cybersecurity topics through direct involvement with labs and exercises. The US Cyber Range team asked the Blacksburg-based graphic design team FourDesign to revamp and redesign the existing website. From this, the FourDesign team reached out to the CS4624 Capstone Computer Science team to research, plan, design, and build it with their more technical knowledge of system architecture and programming skills.

From this, the Computer Science team worked in a multi-level development process in coordination with both the FourDesign team and US Cyber Range. Due to wait times on approvals of certain designs, development processes, and other website components, the Computer Science team also became immersed into the design portion of the website including site mapping, wireframe design, and page content.

The entire design and development process is an effort to keep the website engaging and up to date with information, futuristic appearance, and efficient navigation. This is imperative due to the rapidly developing and expanding realm of Cybersecurity that has become increasingly important in businesses and especially in national defense.

Website development took place initially on an instance of WordPress using a Docker container for independent development. This is due to the inability to gain access to Virginia Tech's cloud services for website development. Further into the website implementation process, this independent development setup using Docker was substituted with XAMPP running Apache and MySQL to host an instance of WordPress. The website was designed specifically on WordPress in order for the Computer Science team to pass the project along to FourDesign and US Cyber Range Team members who are not as proficient with programming. After the semester ends, the clients hope for ease of editing the website where needed. With the given time constraints, the highest priority of implementation for the Computer Science team was developing custom WordPress widgets using a mix of HTML and PHP for specific sections of the approved FourDesign webpage designs. This is so populating the webpages with the widgets and updating website information will require little to no coding for current and future maintainers.

The US Cyber Range Website implementation consisted of:

- Research and education on WordPress development so future developers can easily make changes to content, navigation, and appearance
- Site mapping and planning for efficient and intuitive navigation, concise information, and easily accessible features
- Establishing independent website development environments to adapt to possible access to Virginia Tech cloud-hosting services.
- Page implementation to execute upon the site plans desired by the US Cyber Range and FourDesign Teams.

Introduction

In recent years, computer systems attacks have proven to have the ability to cripple industries, governments, and economies overnight. Consequently, cybersecurity is arguably now a top priority within computer science to prevent and defend against this new evolution of warfare. With multiple world powers competing against one another for technological superiority combined with the ever-present threat of individual malicious hackers, training the masses on these topics and how to use them ethically is a great stride to use these abilities to do good, rather than to create catastrophe.

In 2016, former Virginia governor Terry McAuliffe proposed an effort to expedite the education of computer science, which eventually begat the Virginia Cyber Range centered at Virginia Tech's campus in Blacksburg, VA. After clients outside of the Commonwealth of Virginia expressed their interest in using the Cyber Range's services, the website rebranded as the US Cyber Range [2]. After years of use, the US Cyber Range commissioned the Blacksburg-based graphic design team FourDesign [1] to redesign the website in terms of implementing more intuitive website layout and navigation along with an updated and more appealing website theme to more aptly reflect the purpose of the website.

Subsequently, the FourDesign team assigned the Computer Science 4624 Capstone group consisting of Trenton Deel, Aiden England, Vikramaditya Pratha, and Matthew Rogers to implement the website redesign. A workflow was established consisting of weekly meetings with the FourDesign team, with the US Cyber Range frequently joining as well via Zoom. Upon waiting for the approval of themes and animations created by the FourDesign team, the Computer Science team became immersed in the project design and management aspects of website development through discussion and planning on aspects such as website mapping, navigation, and offering input on proposed designs. Upon approval of designs and wireframes by the US Cyber Range team, the Computer Science team set up instances of WordPress using XAMPP [3], an Apache distribution that includes PHP [6] and MySQL [5] for ease of development workspace setup and management, to support independent website development using the assets provided by the FourDesign team.

The website is being developed through WordPress [4] for ease of transfer to others who will continue to edit and update the website after the Computer Science team's work is done. They should be able to do so without having the prerequisite of advanced coding knowledge. The Computer Science team met weekly with FourDesign to give updates on the implementation of the website wireframes and pages. The final product is a fully redesigned website with more intuitive navigation, clearer information, and appealing themes.

Requirements

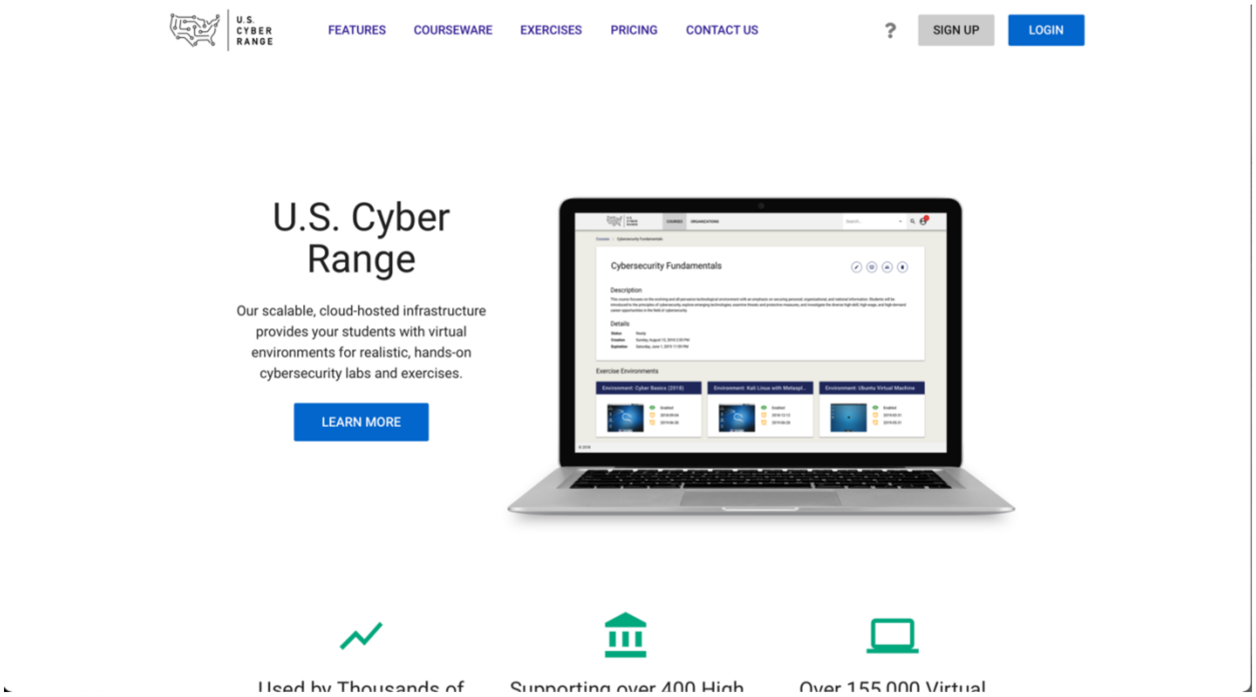

U.S. CYBER RANGE
FEATURES
COURSEWARE
EXERCISES
PRICING
CONTACT US
?
SIGN UP
LOGIN
U.S. Cyber Range
Our scalable, cloud-hosted infrastructure provides your students with virtual environments for realistic, hands-on cybersecurity labs and exercises.
LEARN MORE
Cybersecurity Fundamentals
Used by Thousands of
Supporting over 400 High
Over 155,000 Virtual

Figure 1 Original Website

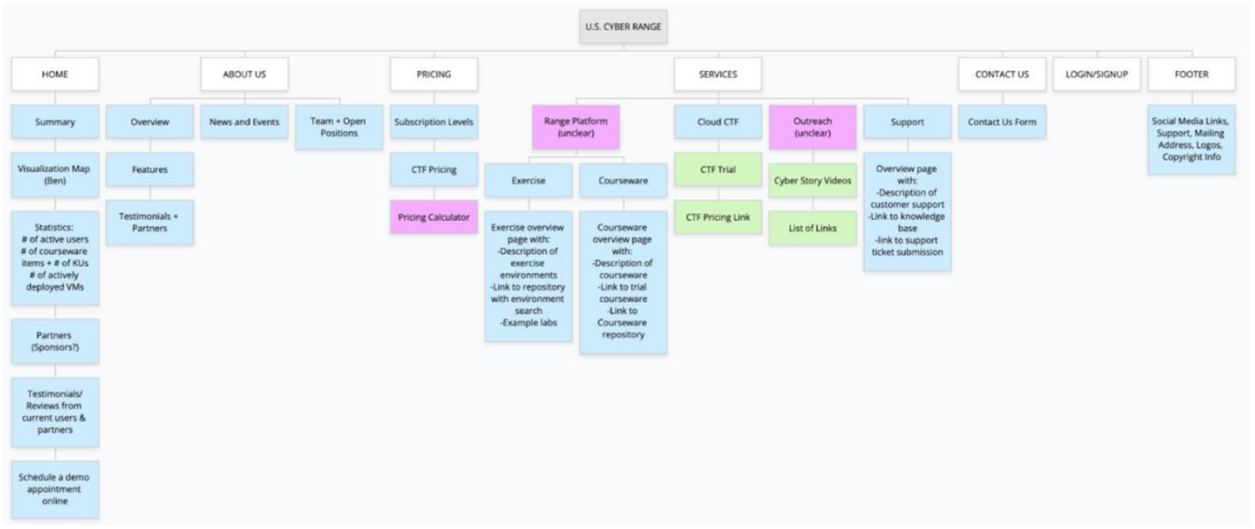
First and foremost, the US Cyber Range client wanted a complete visual redesign of the website. As seen in the above figure the previous US Cyber Range website is bland and lacking the visual appeal that the new website has. The new website had to be visually appealing to both students and teachers. This includes a new website theme, animations, and page layouts.

In addition to the visual component, there are a number of non-visual requirements that had to be met. To ensure that the new and improved designs are seen by a larger population, the client asked that the website be optimized for search engines. With a larger population using the website it is important to maintain accessibility standards. The client specified that the WCAG 2.1 level AA accessibility standards must be met.

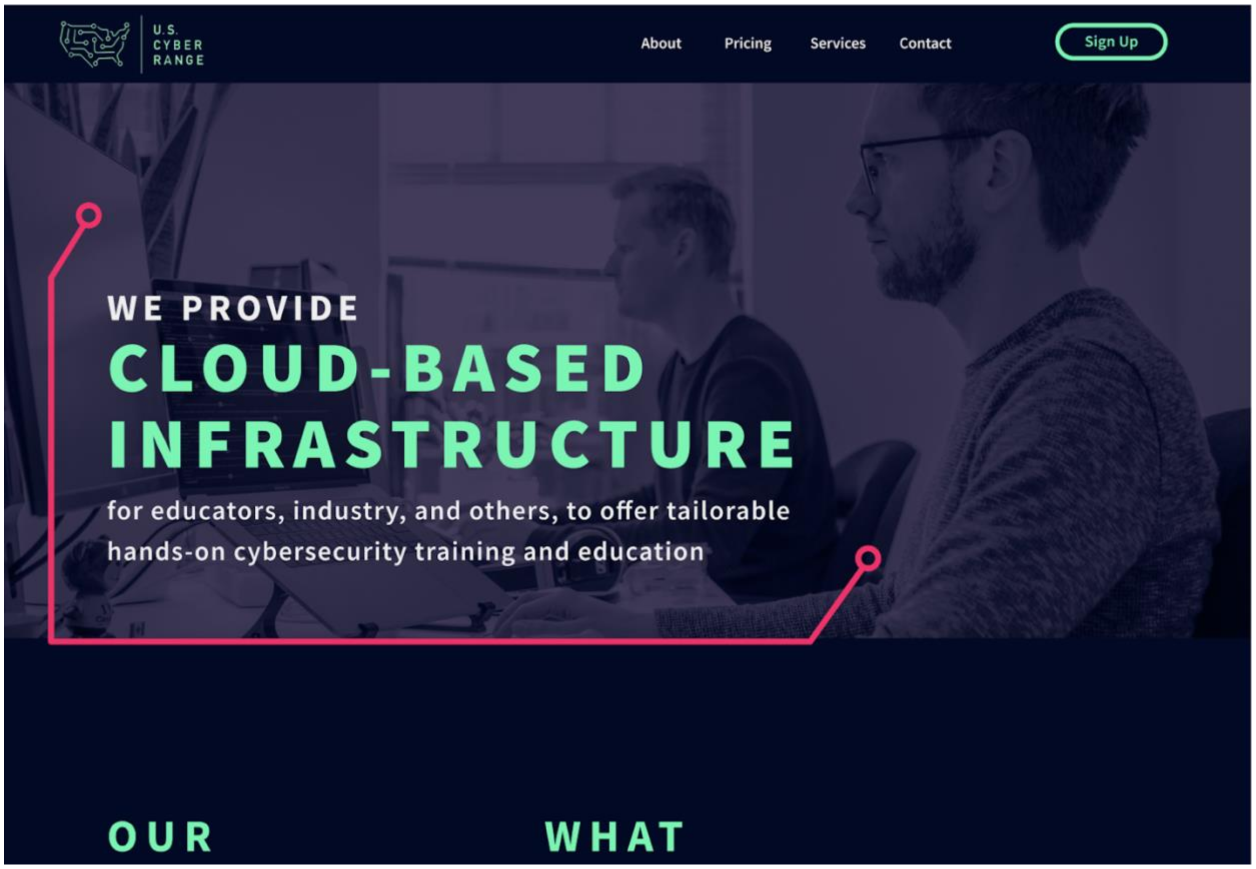
The navigation must also be improved from the previous site. The client stated that, based on their analytics, users could not find desired resources. The navigation improvement consists of rearranging the menu system, changing page names, and adding improved connections between pages. All of these requirements must be met by using a WordPress content management system (CMS) that will allow the website to be maintained by the client. These requirements were outlined in the client proposal [8].

Design

The CS team supported the FourDesign team in creating a new website navigation design. This design follows best practices for page placement. The site map for the full design is shown in Figure 2.


U.S. CYBER RANGE
HOME
ABOUT US
PRICING
SERVICES
CONTACT US
LOGIN/SIGNUP
FOOTER
Summary
Visualization Map (Ben)
Statistics: # of active users # of courseware items + # of KUs # of actively deployed VMs
Partners (Sponsors?)
Testimonials/ Reviews from current users & partners
Schedule a demo appointment online
Overview
Features
Testimonials + Partners
News and Events
Team + Open Positions
Subscription Levels
CTF Pricing
Pricing Calculator
Range Platform (unclear)
Exercise
Exercise overview page with: -Description of exercise environments -Link to repository with environment search -Example labs
Courseware
Courseware overview page with: -Description of courseware -Link to trial courseware -Link to Courseware repository
Cloud CTF
CTF Trial
CTF Pricing Link
Outreach (unclear)
Cyber Story Videos
List of Links
Support
Overview page with: -Description of customer support -Link to knowledge base -link to support ticket submission
Contact Us Form
Social Media Links, Support, Mailing Address, Logos, Copyright Info
Figure 2 Site Map

Each page design, layout and look was developed by the FourDesign team. The head of the about page design is shown in Figure 3.



WE PROVIDE CLOUD-BASED INFRASTRUCTURE for educators, industry, and others, to offer tailorable hands-on cybersecurity training and education

Figure 3 Top of About Page

Implementation

The CS team was solely responsible for implementing the designs created during the design phase of the project. The client specifically requested a CMS system and suggested that the team use WordPress. The implementation follows directly from that requirement. The final product is a customizable WordPress site that looks like the designs.

To create a customizable WordPress site the CS team created a custom WordPress theme. This theme can be easily imported into any WordPress instance. The theme consists of page templates, stylesheets, scripts, widgets, and some WordPress overhead files. This implementation is discussed in detail in the Developer Manual section. However, the general idea is that multiple page template files are created for the client to choose from. These page templates yield pages that look like the designs provided by FourDesign and define widget areas to aid in customizability. A widget area allows the user to select a widget to place in the widget area. A widget is a form the user can fill out, which then outputs customized HTML into the rendered page. Additionally, to adhere to the design, custom widgets were implemented. This allows the user to select between a large number of combinations of widgets and page templates to tweak the design as they see fit. All of this can be done from the admin dashboard without touching the codebase.

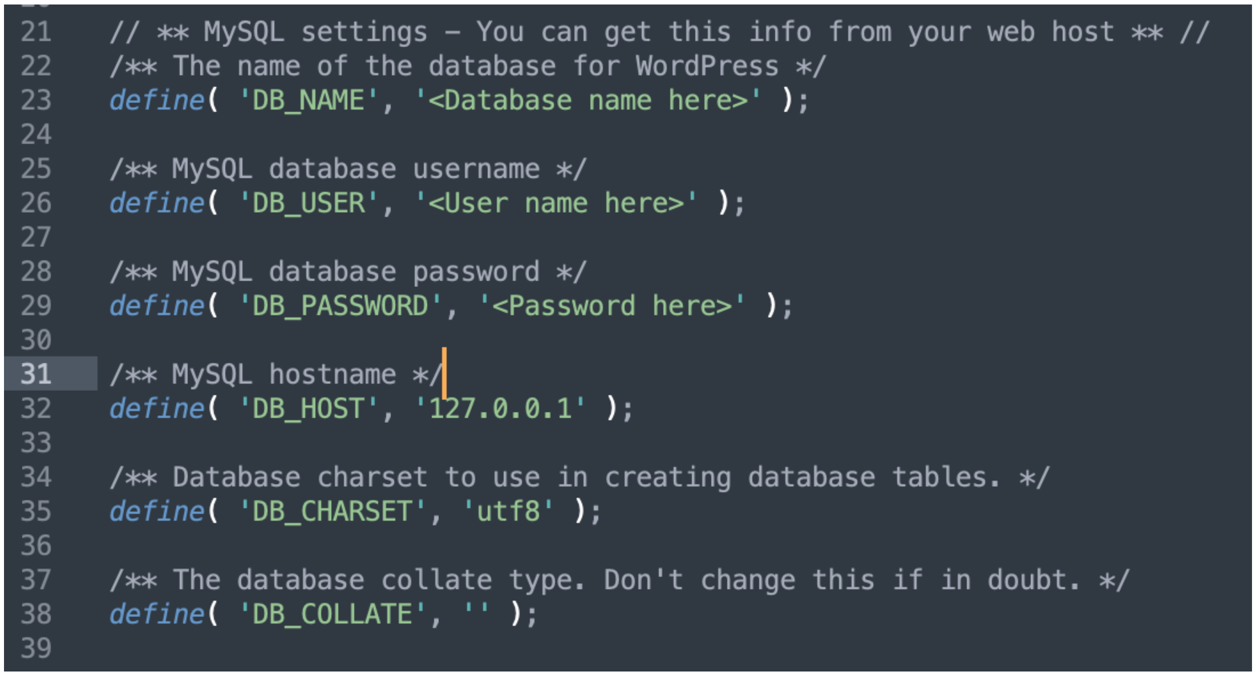
Developer's Manual

Environment Set up

Once the environment is set up, navigation to "<Local WordPress address>/wp-admin" allows access to the admin console. For installation, see the following sections for OS specific information. For further details visit https://wordpress.org/support/article/how-to-install-wordpress/

Mac/Linux

1. Install MySQL and PHP. Versions used by the team are:
   a. mysql Ver 8.0.27 for macos11.6 on x86_64
   b. PHP 7.3.29
2. Set up database:
   a. Create database which will only be used by WordPress.
   b. Create user account for WordPress to use.
3. Install WordPress:
   a. Download from their website.
   b. Unzip files.
   c. Rename wp-config-sample.php to wp-config.php.
   d. Edit wp-config.php to contain database information as in Figure 4.



```php
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', '<Database name here>' );

/** MySQL database username */
define( 'DB_USER', '<User name here>' );

/** MySQL database password */
define( 'DB_PASSWORD', '<Password here>' );

/** MySQL hostname */
define( 'DB_HOST', '127.0.0.1' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );
```

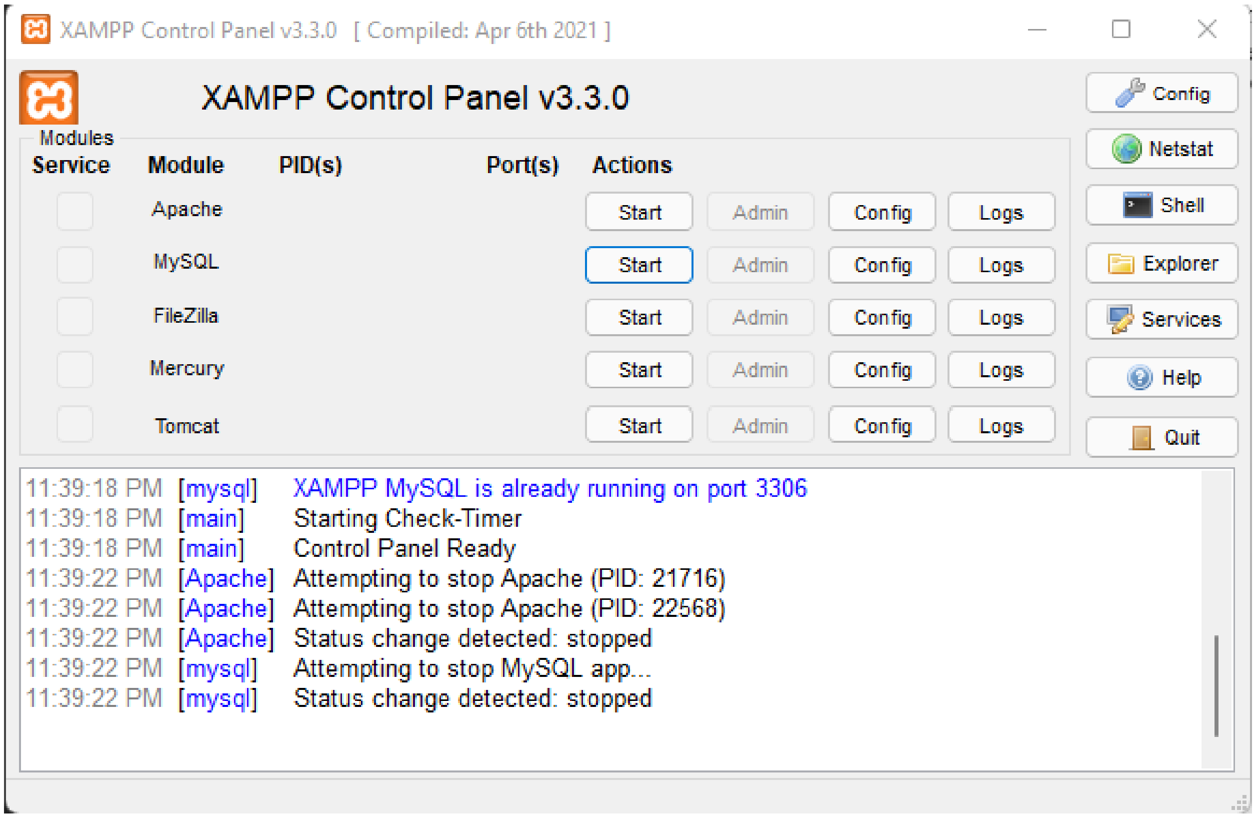
Figure 4 wp-config.php

   e. Navigate to the WordPress directory and run "sudo php -S localhost:80" to start a server.

f. With a browser, open localhost:80 and follow WordPress instructions to finish the setup.
g. Place the custom theme folder in the WordPress directory/wp-content/themes/.
h. Set custom theme as active.

Windows

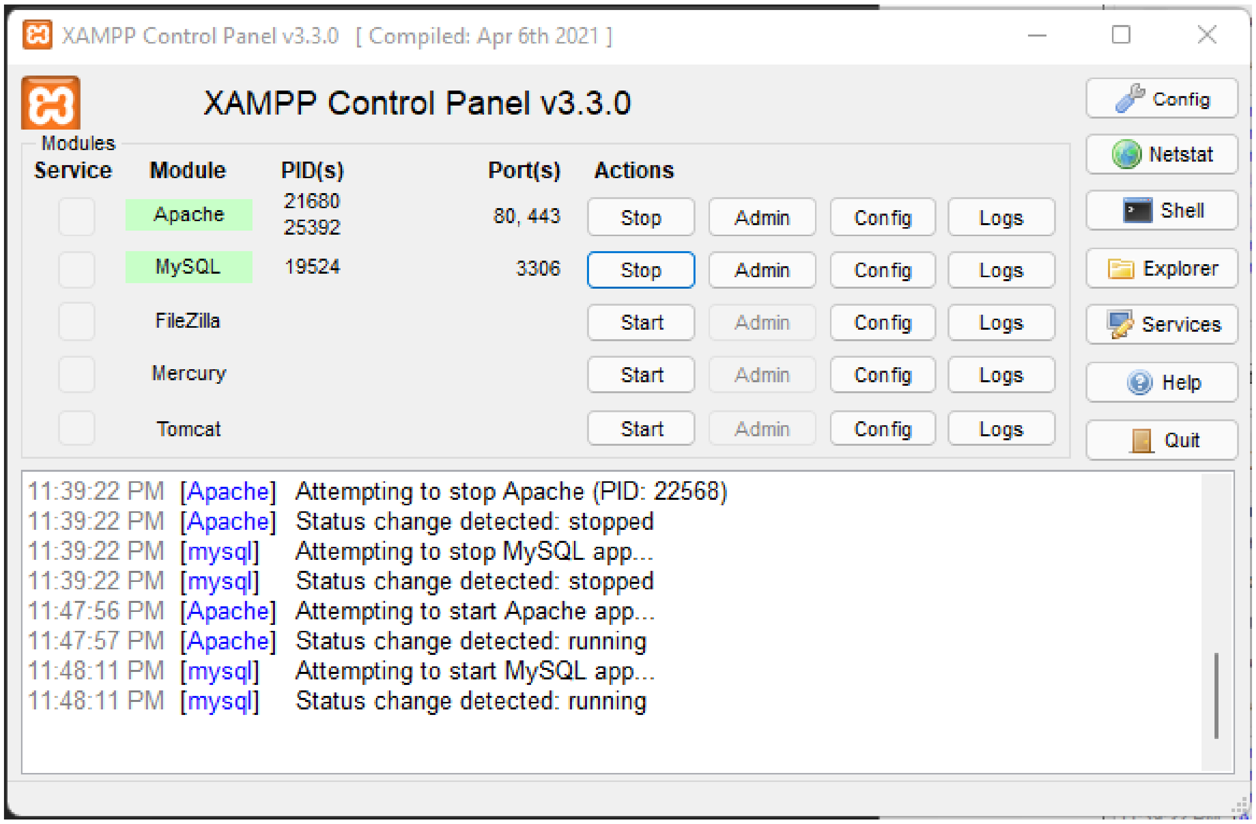
1. Install XAMPP [3] 64-bit from https://www.apachefriends.org/download.html for your corresponding operating system in the default locations (C:\xampp for Windows OS)
    1. Make sure to install Apache, MySQL, and PHP packages within XAMPP installer
2. Start the XAMPP dashboard.
3. Start Apache and MySQL from the XAMPP dashboard, as shown in Figure 5.



```
11:39:18 PM  [mysql]   XAMPP MySQL is already running on port 3306
11:39:18 PM  [main]    Starting Check-Timer
11:39:18 PM  [main]    Control Panel Ready
11:39:22 PM  [Apache]  Attempting to stop Apache (PID: 21716)
11:39:22 PM  [Apache]  Attempting to stop Apache (PID: 22568)
11:39:22 PM  [Apache]  Status change detected: stopped
11:39:22 PM  [mysql]   Attempting to stop MySQL app...
11:39:22 PM  [mysql]   Status change detected: stopped
```

Figure 5 XAMPP Initial Start

4. Click the “Admin” Button for MySQL on the XAMPP dashboard.

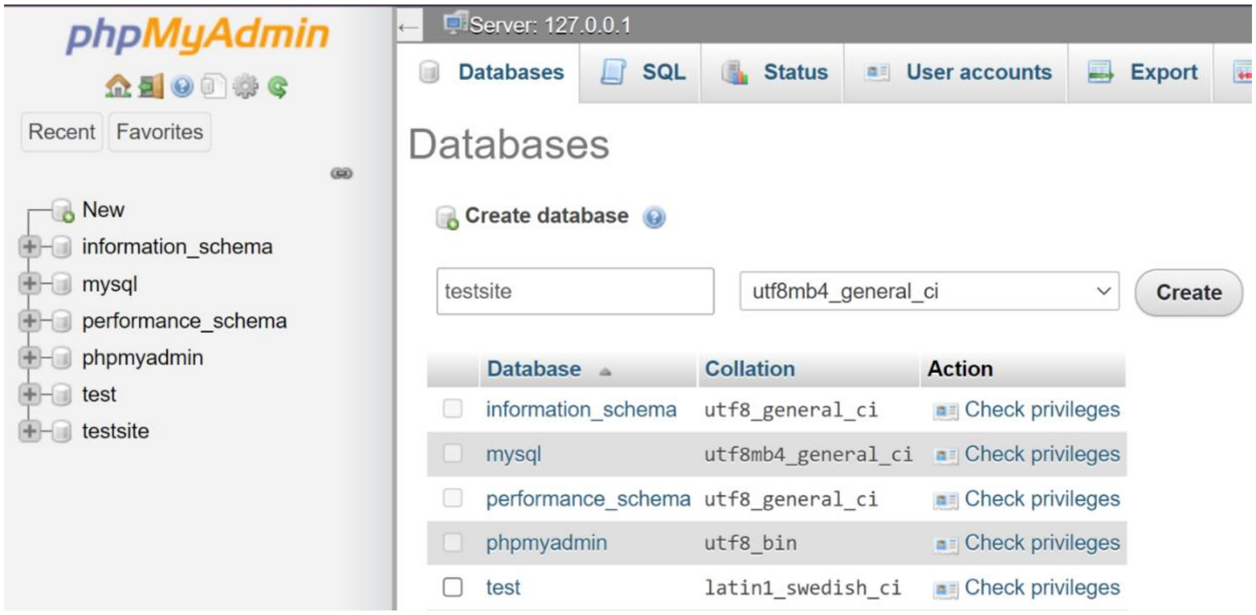

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	21680 25392	80, 443	Stop	Admin	Config	Logs
	MySQL	19524	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
11:39:22 PM  [Apache]  Attempting to stop Apache (PID: 22568)
11:39:22 PM  [Apache]  Status change detected: stopped
11:39:22 PM  [mysql]   Attempting to stop MySQL app...
11:39:22 PM  [mysql]   Status change detected: stopped
11:47:56 PM  [Apache]  Attempting to start Apache app...
11:47:57 PM  [Apache]  Status change detected: running
11:48:11 PM  [mysql]   Attempting to start MySQL app...
11:48:11 PM  [mysql]   Status change detected: running
```

Figure 6 XAMPP MYSQL Start

5. Create new SQL server named “testsite”
a. Keep username and password blank

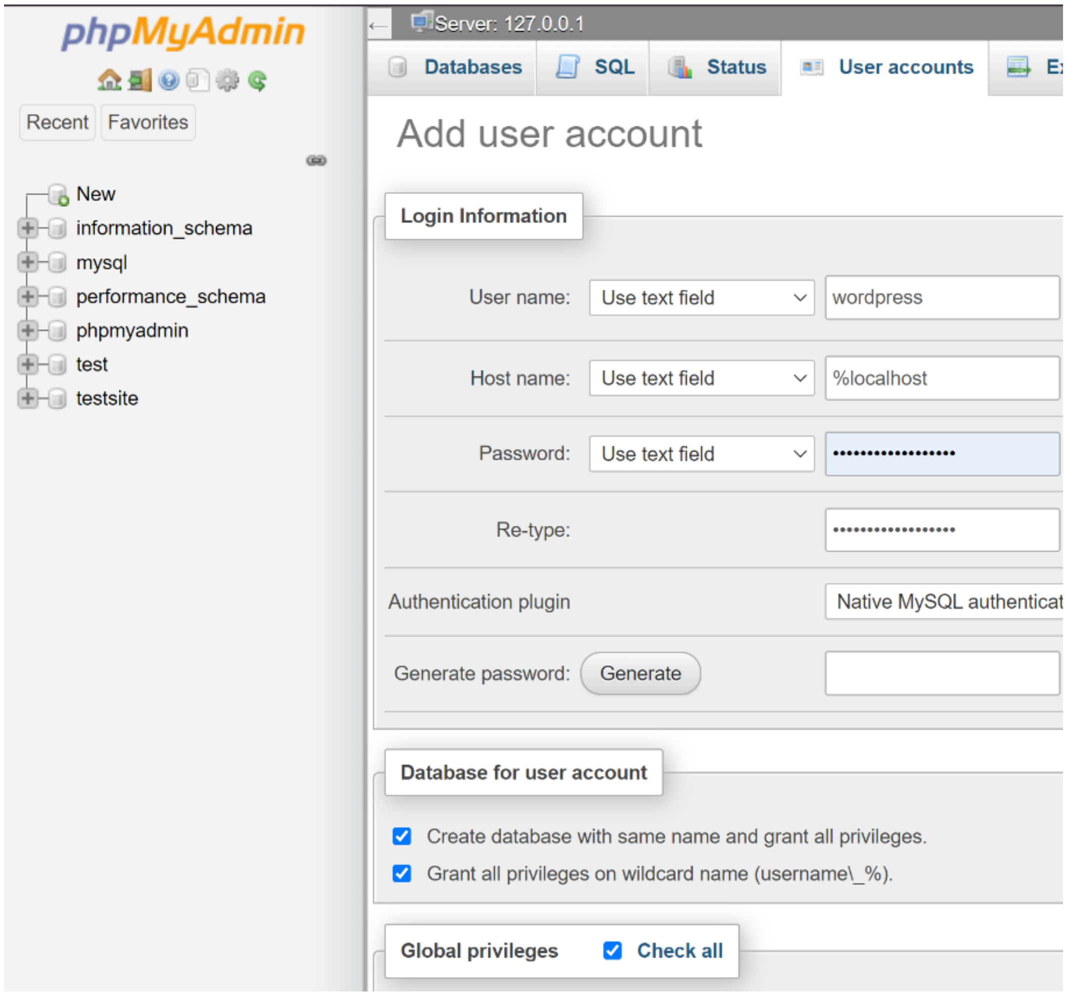


testsite | utf8mb4_general_ci

Database	Collation	Action
information_schema	utf8_general_ci	Check privileges
mysql	utf8mb4_general_ci	Check privileges
performance_schema	utf8_general_ci	Check privileges
phpmyadmin	utf8_bin	Check privileges
test	latin1_swedish_ci	Check privileges

Figure 7 XAMPP Database Creation

6. Click "Create" to create a new database.
7. In phpMyAdmin, navigate to the "User accounts" tab.
    1. Under "User accounts overview", click "Add user account" in the gray container.
    2. When creating a new user:
        1. Username: wordpress
        2. Hostname: %localhost
        3. Create and retype password
        4. Check both boxes under "Database for user account"
        5. Check "Check all" next to "Global privileges"



User name: Use text field | wordpress

Host name: Use text field | %localhost

Password: Use text field | ••••••••••••••••••

Re-type: ••••••••••••••••••

Authentication plugin: Native MySQL authenticat

Generate password:

☑ Create database with same name and grant all privileges.

☑ Grant all privileges on wildcard name (username_%).

☑ Check all

Figure 8 XAMPP User Creation

8. Click "Go" at the bottom of the page.
9. Download WordPress from https://wordpress.org/download/#download-install
    1. Create a free WordPress account if needed.
    2. Download results will have a Zip folder type.

10. Extract the WordPress distribution to C:\xampp\htdocs\wordpress (or wherever the XAMPP folder was installed if not in the aforementioned location).

11. Rename C:\xampp\htdocs\wordpress to C:\xampp\htdocs\wordpress. Make sure the next containing folder has the directories wp-contents, wp-admin, and wp-includes. Otherwise, remove the extra directory layer before the aforementioned directories.
12. Rename C:\xampp\htdocs\testsite\wp-config-sample.php to C:\xampp\htdocs\testsite\wp-config.php
13. Open C:\xampp\htdocs\testsite\wp-config.php
    1. Change the attribute 'DB_NAME' to 'testsite'
    2. Change the attribute 'DB_USER' to 'wordpress'
    3. Change the attribute 'DB_PASSWORD' to 'admin'
    4. Change the attribute 'DB_HOST' to 'localhost
14. Open a browser and type localhost/testsite/wp-admin in the search bar.
15. Establish and write down credentials in required fields in WordPress Login
    1. This will initiate the WordPress installation.
16. Download Git Bash
17. In the browser, navigate to the theme repository at git.cs.vt.edu
18. Locate CyberRangeCustomTheme project
19. In the project, click the down arrow on "Clone" and select the clipboard button on "Clone with HTTPS".

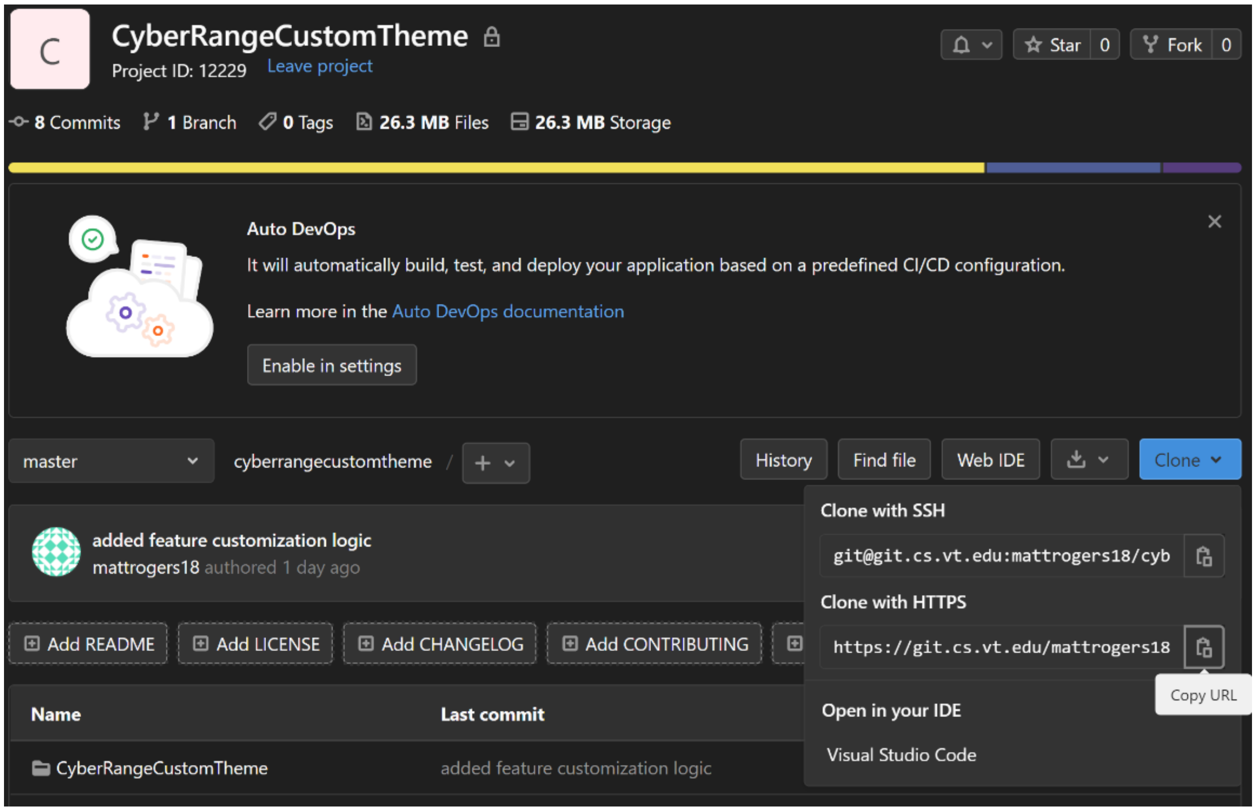


Figure 9 Repository

20. Open Git Bash
21. If in Windows, type "cd ../.." and press ENTER.

22. Type “git clone “, paste the “Clone with HTTPS” command copied in step 19, and press ENTER. This should clone the theme repository to your local machine and give it the name “cyberrangecustomtheme”.

23. In Git bash, copy the theme folder to your WordPress folder using the command “cp -r ./cyberrangecustomtheme/CyberRangeCustomTheme ./xampp/htdocs/testsite/wp-content”

24. Download any IDE (preferably Visual Studio Code), and open the CyberRangeCustomTheme copy folder within your xampp folder.

25. In Visual Studio Code, make sure to install any suggested software packages from the IDE (most likely PHP).

File Structure

Within the custom theme folder, the file structure is as given in Figure 10.

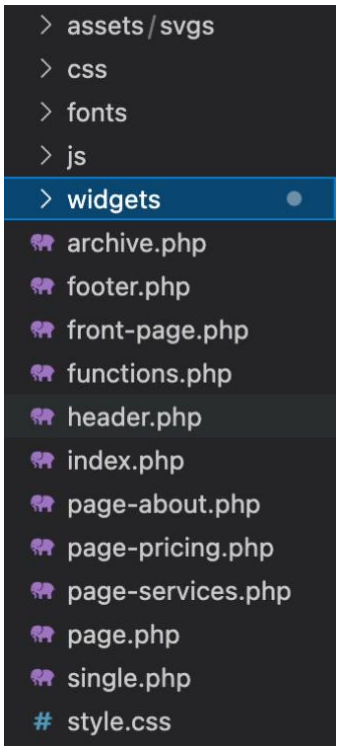

```
> assets / svgs
> css
> fonts
> js
> widgets
archive.php
footer.php
front-page.php
functions.php
header.php
index.php
page-about.php
page-pricing.php
page-services.php
page.php
single.php
# style.css
```

Figure 10 File Structure

Each folder/file is used as follows

- Assets folder
    - Contains any assets for the page such as files with suffix .svg, .img, or .gif.
- CSS folder
    - Contains CSS files
    - Also contains BootstrapV4
- Fonts folder
    - Font files necessary to create the font-faces used.
- Js folder

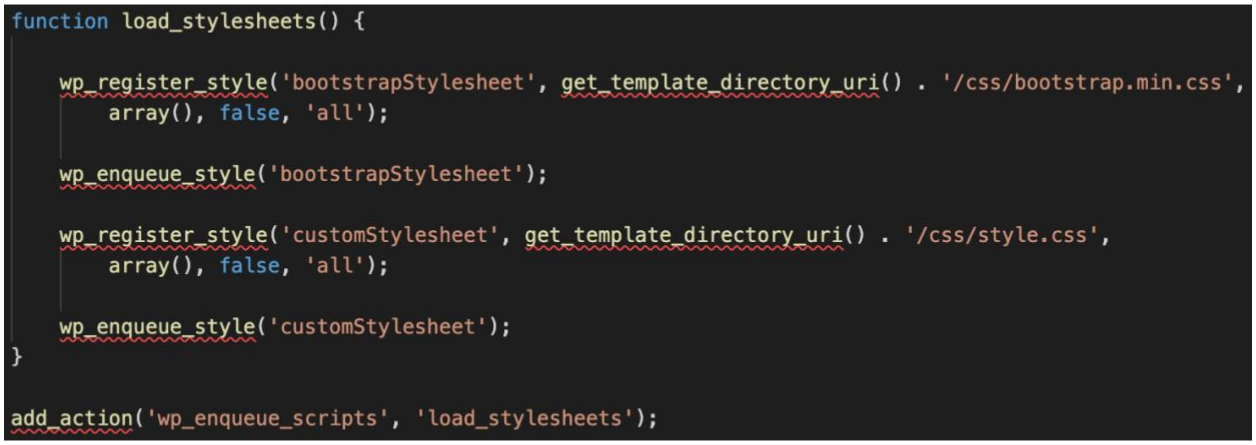
    - Contains all scripts used
    - Also contains JQueryV3.6.0
- Widgets folder
    - Contains all created widget files. A further break down of the widget file structure is found in the widget section.
- Style.css
    - This is a file with a comment necessary for WordPress to load the theme.
- Functions.php
    - Important file for WordPress to use the theme. Further explained in the functions.php section.
- Other (All files not displayed in Figure 10 not already mentioned)
    - Will be explained in Page Structure section

Functions.php

WordPress uses this file to load the theme. Functions in this file load scripts, stylesheets, widgets, and widget areas. Loading scripts and stylesheets will be discussed here but loading widgets/widget-areas will be described in the widget section.

Loading Stylesheets

Some style sheets are already being loaded as seen in the picture below. Explanation of how this works is given in Figure 11.



```
function load_stylesheets() {

    wp_register_style('bootstrapStylesheet', get_template_directory_uri() . '/css/bootstrap.min.css',
        array(), false, 'all');

    wp_enqueue_style('bootstrapStylesheet');

    wp_register_style('customStylesheet', get_template_directory_uri() . '/css/style.css',
        array(), false, 'all');

    wp_enqueue_style('customStylesheet');
}

add_action('wp_enqueue_scripts', 'load_stylesheets');
```

Figure 11 Loading Stylesheets

1. A function is created in functions.php. In this example it is called “load_stylesheets”.
2. For each stylesheet do the following inside the created function (load_stylesheets).
    i) Use wp_register_style to register the stylesheet. Wp_register_style takes arguments listed below.
        (1) A name of the stylesheet. This name is only used in enqueueing the stylesheets
        (2) A location of the stylesheet. Use get_template_directory_uri() to get the base directory of the theme, then tack on the additional structure to get to the stylesheet

(3) The last three arguments can follow exactly as seen above specifically “array(), false, ‘all’”. These define basic behavior. If you wish to change it, visit the WordPress documentation for details.

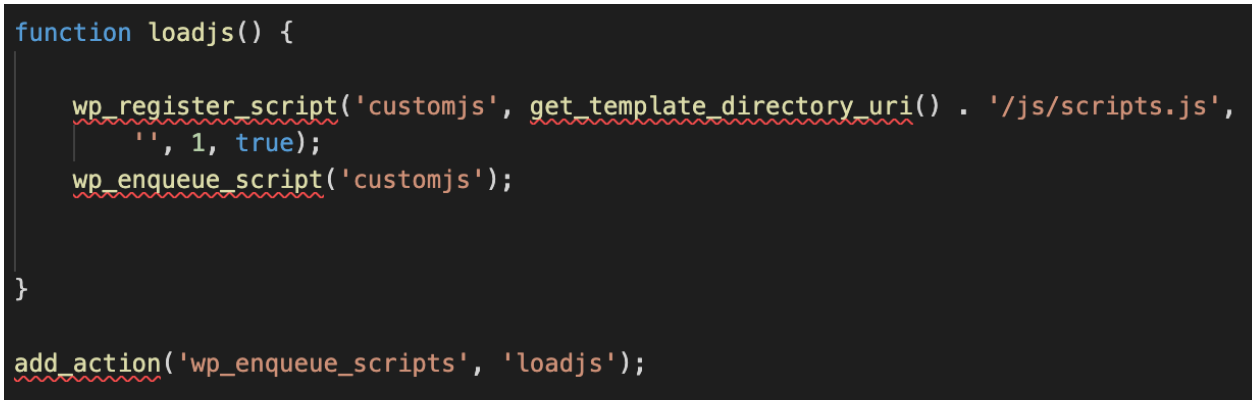
ii) Use wp_enqueue_style to load the stylesheet into WordPress. Provide the name of the stylesheet created with wp_register_style to the function.

3. To get WordPress to run this function add_action as displayed in Figure 11 is used.

A general note on styles: Most styles can be pulled directly from the Figma [7]file provided by the FourDesign team. The Figma file displays CSS when inspecting an element.

Loading Scripts

Loading scripts is essentially the same process as loading stylesheets. The differences are minor and can be seen in Figure 12. Notice the different wp_ function names.

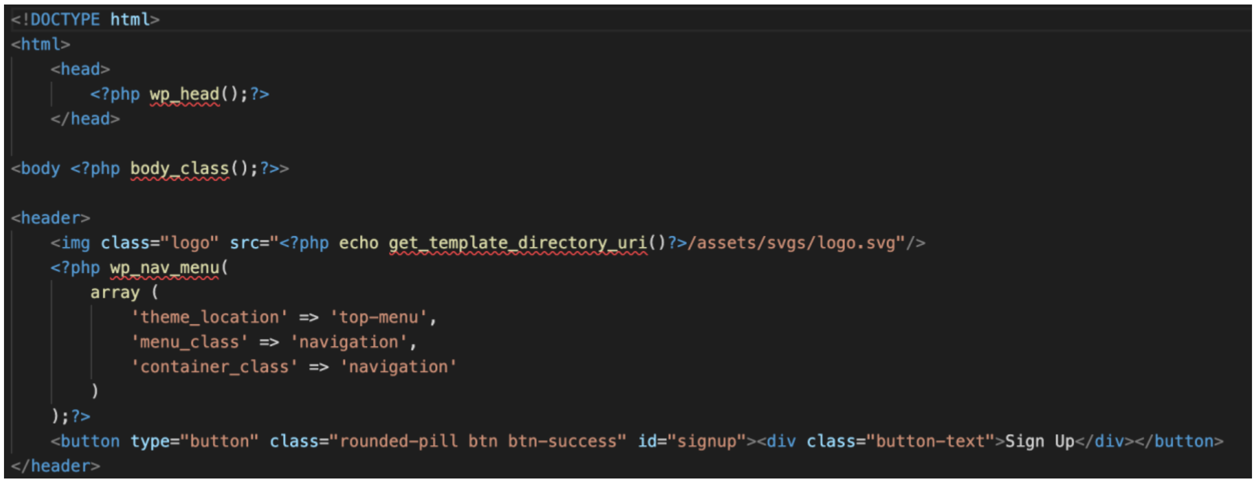


```
function loadjs() {

    wp_register_script('customjs', get_template_directory_uri() . '/js/scripts.js',
        '', 1, true);
    wp_enqueue_script('customjs');


}

add_action('wp_enqueue_scripts', 'loadjs');
```

Figure 12 Loading Scripts

Page Structure

Header/Footer.php



```
<!DOCTYPE html>
<html>
    <head>
        <?php wp_head();?>
    </head>

<body <?php body_class();?>>

<header>
    <img class="logo" src="<?php echo get_template_directory_uri()?>/assets/svgs/logo.svg"/>
    <?php wp_nav_menu(
        array (
            'theme_location' => 'top-menu',
            'menu_class' => 'navigation',
            'container_class' => 'navigation'
        )
    );?>
    <button type="button" class="rounded-pill btn btn-success" id="signup"><div class="button-text">Sign Up</div></button>
</header>
```

Figure 13 header.php

Figure 13 displays the whole header.php file. This was created to be included on every page. The specifics on how to do that are in the Page-*.php section. The goal of the header.php file is to set up the page with opening HTML and body tags, as well as provide any content that should be seen on every page. Currently the header only adds a logo, menu system, and sign-up button.

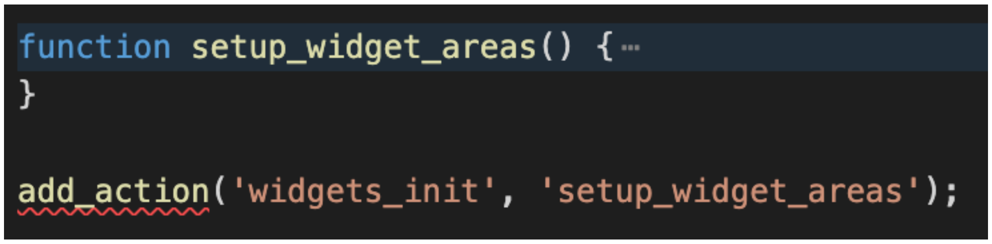
Similarly, the footer (not shown) closes the tags and adds any content that should be viewed at the bottom of every page. When using both the header and footer with a page template the resulting page is going to be a concatenation of the three files (header + page + footer).

Page-*.php

Notice the * in the section title. This is the format for all custom page templates, except for the front-page. The * can be replaced with any word and represents a slug. A slug in WordPress allows the user to select which page template to use when creating a new page. For example, the about page, with a file name of page-about.php, allows the user to type "about" in a slug field to use that page template. If no slug is provided by the user, then the page.php file is used instead. This file should contain an all-purpose page template.

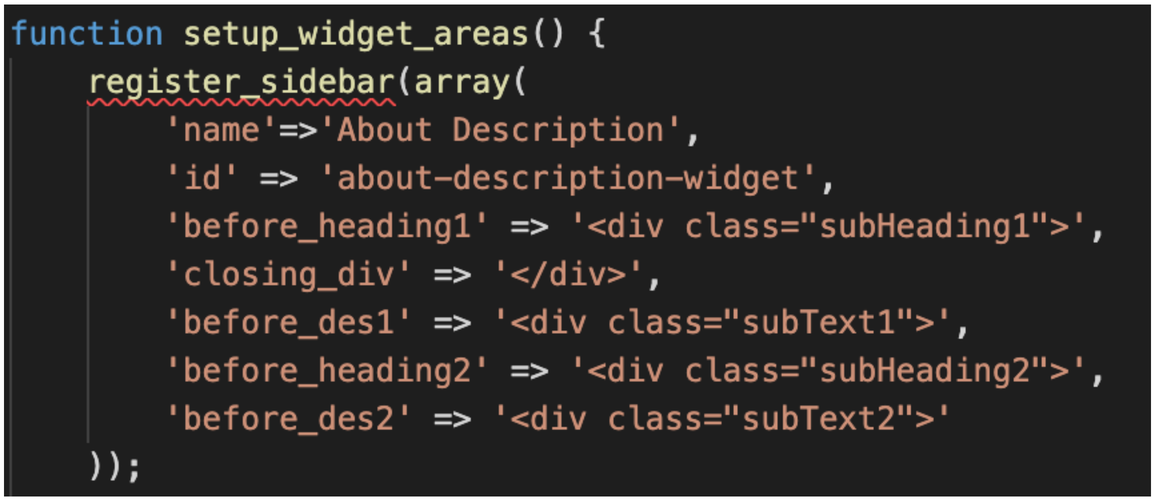
Each page file contains the HTML/PHP that will be displayed on the front end. Each page has a few notable features.

- Header/Footer include
    - The first line should include the header with "<?php get_header(); ?>"
    - The last line should include the footer with "<?php get_footer(); ?>"
- Widget areas
    - A widget area is a designated place on the page that allows the user to place a widget.
    - They are created by placing the following statement somewhere in the template file: "<?php dynamic_sidebar('widget_area_name');?>"
    - These widget areas must be registered in the functions.php file as seen in Figure 14.



```
function setup_widget_areas() {⋯
}

add_action('widgets_init', 'setup_widget_areas');
```

Figure 14 Widget Area Function



```
function setup_widget_areas() {
    register_sidebar(array(
        'name'=>'About Description',
        'id' => 'about-description-widget',
        'before_heading1' => '<div class="subHeading1">',
        'closing_div' => '</div>',
        'before_des1' => '<div class="subText1">',
        'before_heading2' => '<div class="subHeading2">',
        'before_des2' => '<div class="subText2">'
    ));
```

Figure 15 Widget Area Registration Example

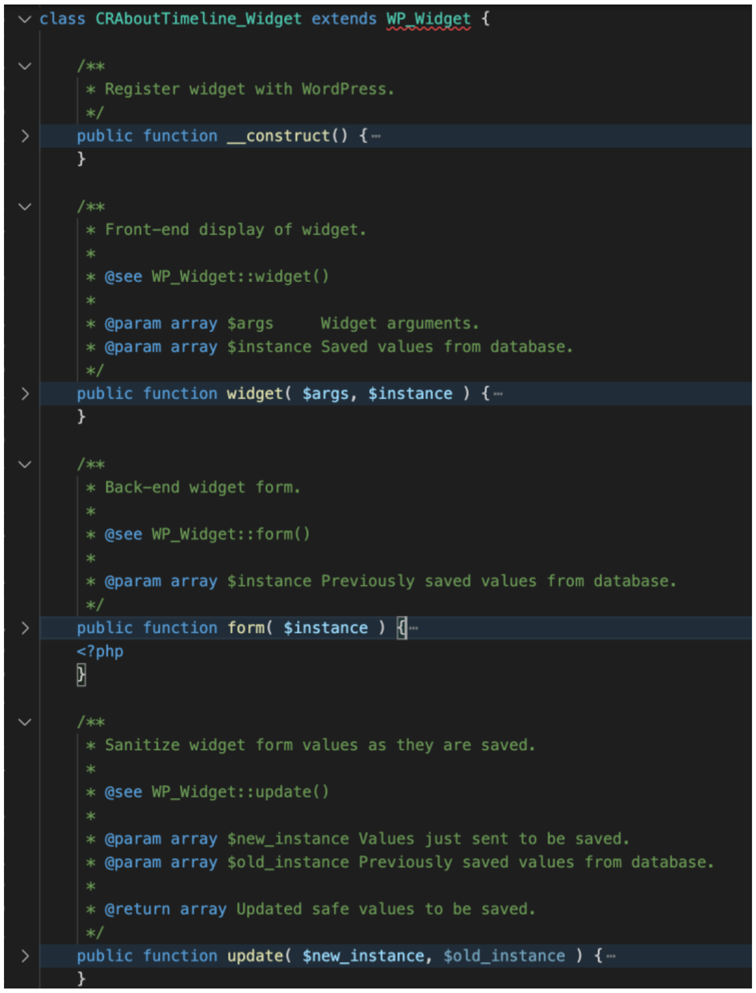
- The widget areas are currently registered in a function called setup_widget_areas which is called by add_action with the widgets_init param. See Figure 15.
- The name and ID parameters are required for the register_sidebar array.
  - Name is the name the user will see when selecting the widget area
  - id should be the name that appears in the dynamic_sidebar function in the page template file.
  - The other optional parameters are given to the widget in an $args array which will be discussed further in the widget section
    - Any page specific HTML needed for the widget to render properly should be placed in these extra arguments.

Front-page.php

This can be created the same as the page-*.php files but is special because it defines the landing page of the website.

Widget Structure

A widget can be placed in a page's widget area. WordPress has a number of built-in widgets; however, it may be necessary to create custom widgets. A widget contains four important sections. These sections can be seen in Figure 16 and will be explained separately.

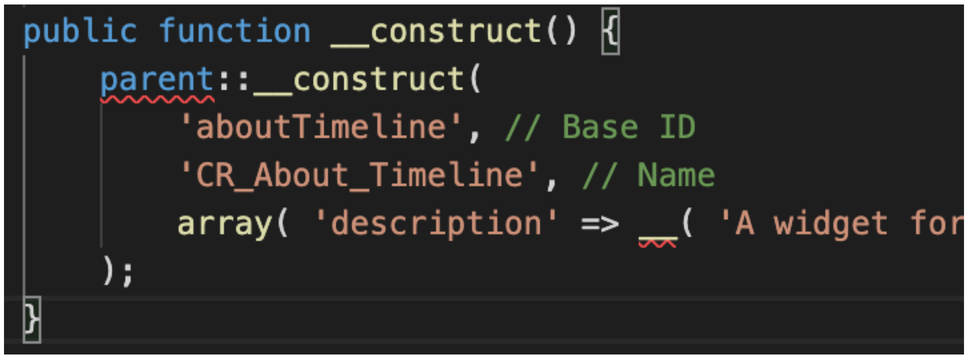


```
class CRAboutTimeline_Widget extends WP_Widget {

    /**
     * Register widget with WordPress.
     */
    public function __construct() {…
    }

    /**
     * Front-end display of widget.
     *
     * @see WP_Widget::widget()
     *
     * @param array $args     Widget arguments.
     * @param array $instance Saved values from database.
     */
    public function widget( $args, $instance ) {…
    }

    /**
     * Back-end widget form.
     *
     * @see WP_Widget::form()
     *
     * @param array $instance Previously saved values from database.
     */
    public function form( $instance ) {…
    <?php
    }

    /**
     * Sanitize widget form values as they are saved.
     *
     * @see WP_Widget::update()
     *
     * @param array $new_instance Values just sent to be saved.
     * @param array $old_instance Previously saved values from database.
     *
     * @return array Updated safe values to be saved.
     */
    public function update( $new_instance, $old_instance ) {…
    }
```

Figure 16 Widget Structure

Constructor

First the widget contains a class name and should extend the WP_Widget class. From there the widget has a __construct method WordPress uses to create the widget. The method should include at least what is shown in Figure 17.



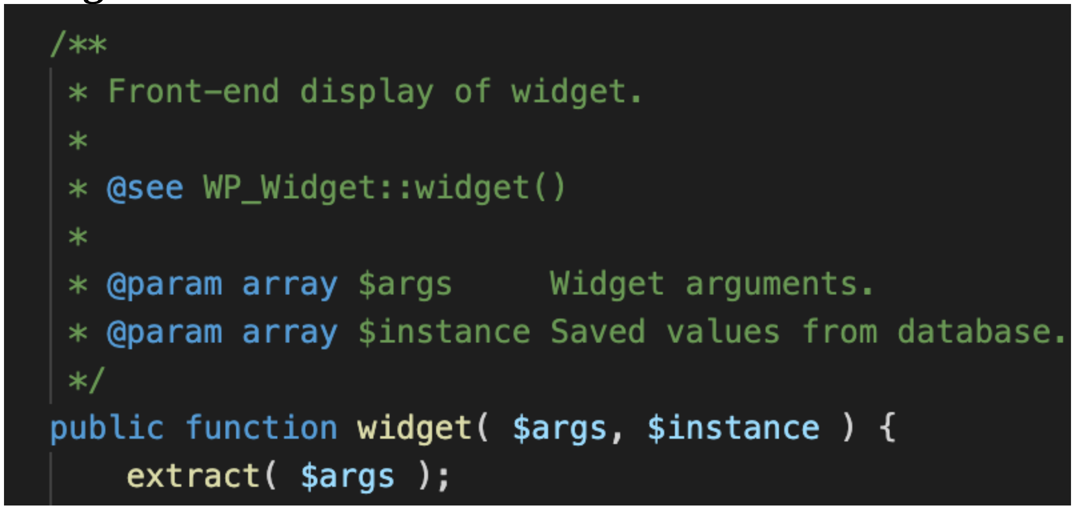
```
public function __construct() {
    parent::__construct(
        'aboutTimeline', // Base ID
        'CR_About_Timeline', // Name
        array( 'description' => __( 'A widget for
    );
}
```

Figure 17 Widget Constructor

- A base id

- Name
    - This is the name of the widget shown to the user. Currently all custom widgets designed by the team are prefixed with CR_
- A description formatted as in Figure 17. This is a description the user will see when inspecting the widget from the WordPress admin dashboard.

Widget function

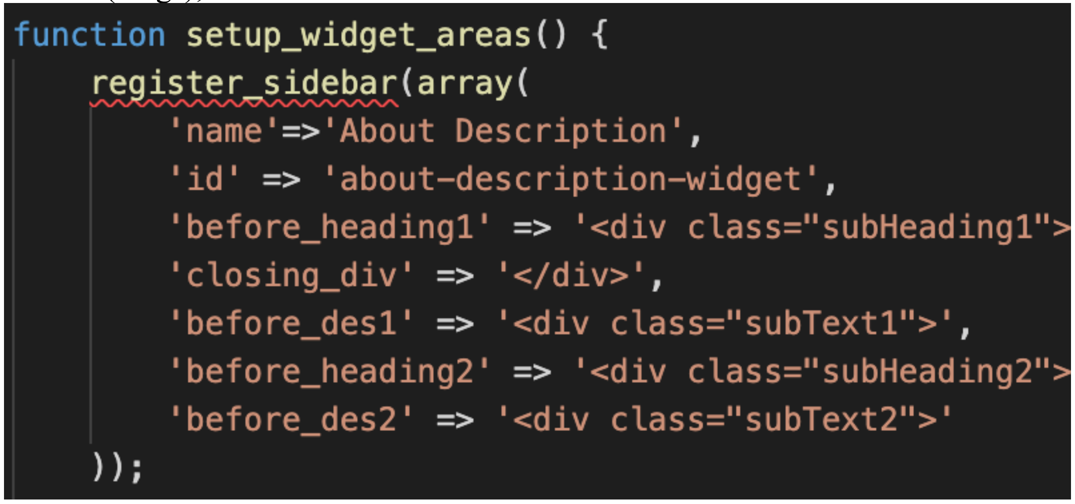


```
/**
 * Front-end display of widget.
 *
 * @see WP_Widget::widget()
 *
 * @param array $args     Widget arguments.
 * @param array $instance Saved values from database.
 */
public function widget( $args, $instance ) {
    extract( $args );
```

Figure 18 Widget Function

The widget function creates the HTML/PHP that will be included in the rendered page. It does this by using PHP's echo function to echo HTML into the page. The arguments provided are $args and $instance.

The $args argument is an array created by WordPress with the values from the widget area the widget is being placed into. That means that the values defined in functions.php can be used after "extract($args);" is called.



```
function setup_widget_areas() {
    register_sidebar(array(
        'name'=>'About Description',
        'id' => 'about-description-widget',
        'before_heading1' => '<div class="subHeading1">
        'closing_div' => '</div>',
        'before_des1' => '<div class="subText1">',
        'before_heading2' => '<div class="subHeading2">
        'before_des2' => '<div class="subText2">'
    ));
```

Figure 19 $args Source

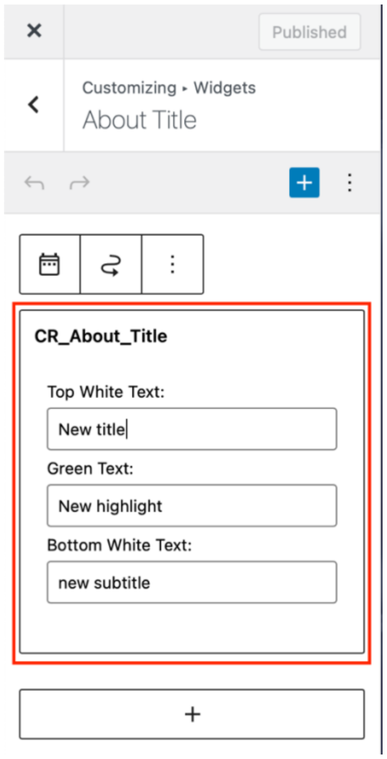
With the widget area defined in Figure 19, placing the statement “echo $before_heading1” in the widget function that is being placed in the about-description-widget area will create <div class=”subHeading1”> on the rendered page.

Additionally, the $instance variable contains all of the fields set by the user. These fields are created in the form function. The $instance variable is an array. Here is a quick example of how it would be used to create a heading assuming the form function created an item with key ‘heading’:

“echo “<h1>” . $instance[‘heading’] . “</h1>”

Form function

The form function defines what the user will see when selecting to use the widget and what values the user can set. It defines what can be seen when the user is customizing their site. It is shown highlighted in red in Figure 20.



CR_About_Title

Top White Text: New title

Green Text: New highlight

Bottom White Text: new subtitle

Figure 20 User View of Widget Form

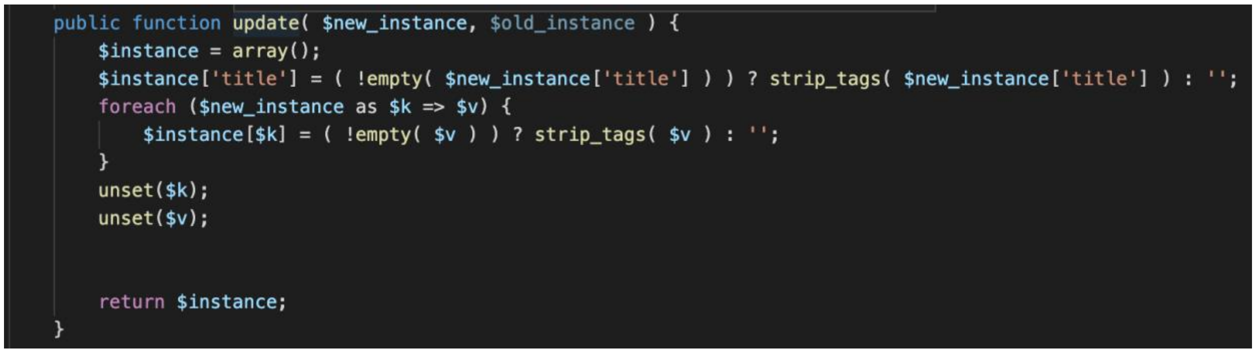
To create this input field the form function allows the developer to switch between HTML and PHP. Any HTML within this function will be displayed to the user. This allows any form to be created. An important note is that every input area should have some key attributes as in:

```
<input class="widefat" id="<?php echo $this->get_field_id( 'title' ); ?>" name="<?php echo $this->get_field_name( 'title' ); ?>" type="text" value="<?php echo esc_attr( $instance['title'] ); ?>"/>
```

To set the $instance array with an element the following fields have to be included in the input area

- Id
  - id="<?php echo $this->get_field_id('name); ?>"
- Name
  - name="<?php echo $this->get_field_name('name'); ?>"
- value
  - value="<?php echo esc_attr($instance['name']);
  - This is the actual value that will reside in $instance['name']

Update function

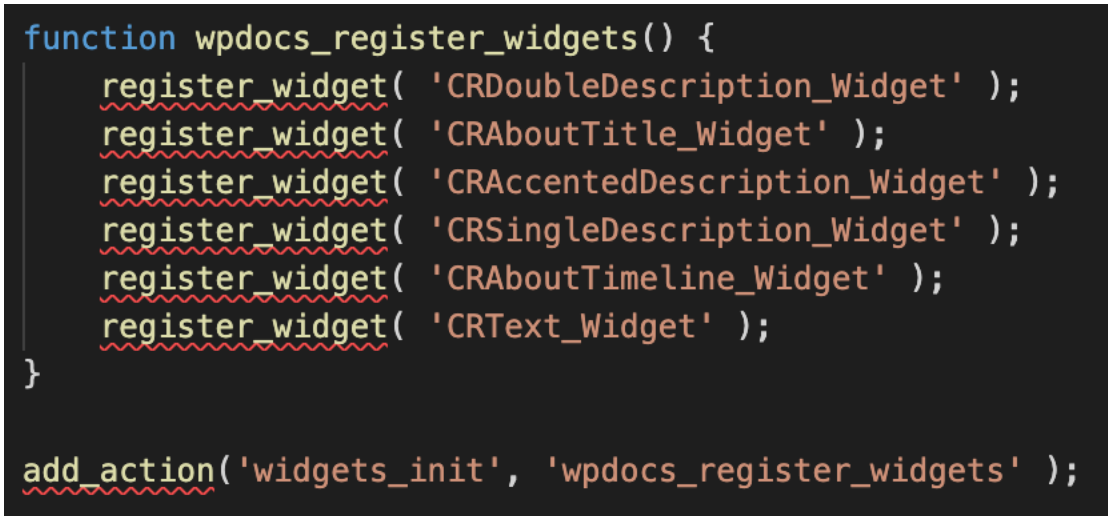


```
public function update( $new_instance, $old_instance ) {
    $instance = array();
    $instance['title'] = ( !empty( $new_instance['title'] ) ) ? strip_tags( $new_instance['title'] ) : '';
    foreach ($new_instance as $k => $v) {
        $instance[$k] = ( !empty( $v ) ) ? strip_tags( $v ) : '';
    }
    unset($k);
    unset($v);


    return $instance;
}
```

Figure 21 Widget Update Function

This function is just for housekeeping. The idea is to strip tags off the current variables in the instance. It is called when a widget is updated. Most widgets can just follow the format shown in Figure 21.

Register Widget



```
function wpdocs_register_widgets() {
    register_widget( 'CRDoubleDescription_Widget' );
    register_widget( 'CRAboutTitle_Widget' );
    register_widget( 'CRAccentedDescription_Widget' );
    register_widget( 'CRSingleDescription_Widget' );
    register_widget( 'CRAboutTimeline_Widget' );
    register_widget( 'CRText_Widget' );
}

add_action('widgets_init', 'wpdocs_register_widgets' );
```

Figure 22 Registering Widgets

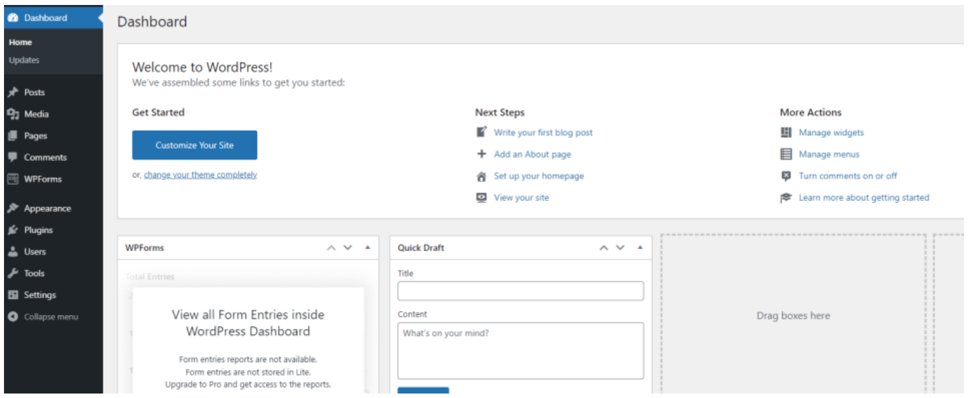
To use the custom widgets from the WordPress admin dashboard they must be registered in functions.php as seen in Figure 22.

User’s Manual

Importing the Website Theme

The custom theme is able to be imported into WordPress as a zip file through the WordPress dashboard. A user can also add the custom theme to their website by placing the custom theme folder in the theme folder of the WordPress instance.

Selecting the Website Theme



We've assembled some links to get you started:

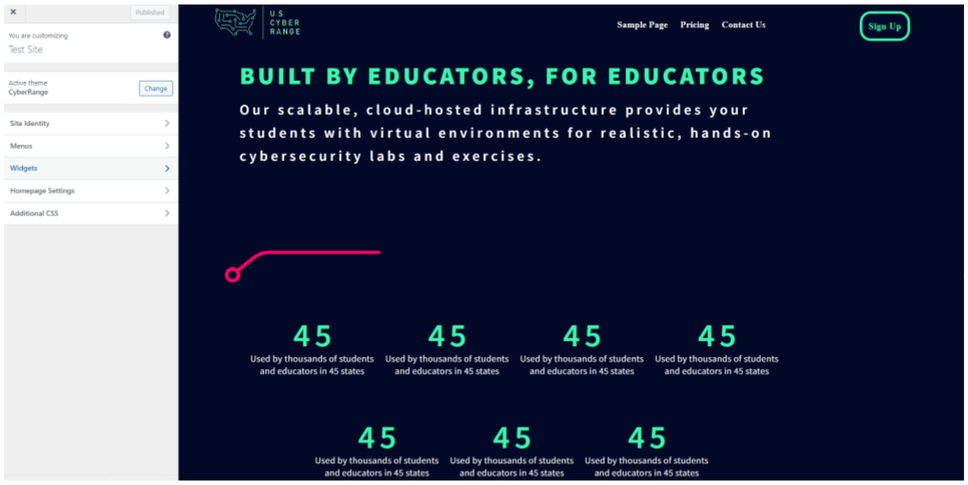
View all Form Entries inside WordPress Dashboard

Form entries reports are not available.
Form entries are not stored in Lite.
Upgrade to Pro and get access to the reports.

Figure 23 WordPress Admin Dashboard

Changing the theme of the WordPress site to the CyberRangeCustomTheme is possible by navigating to the Dashboard page of your site, shown in Figure 23, and choosing “change your theme completely”. Here, you will be presented with a choice of sample themes and the custom theme that our team has created.

Using Widgets



BUILT BY EDUCATORS, FOR EDUCATORS

Our scalable, cloud-hosted infrastructure provides your students with virtual environments for realistic, hands-on cybersecurity labs and exercises.

45 Used by thousands of students and educators in 45 states

45 Used by thousands of students and educators in 45 states

45 Used by thousands of students and educators in 45 states

45 Used by thousands of students and educators in 45 states

45 Used by thousands of students and educators in 45 states

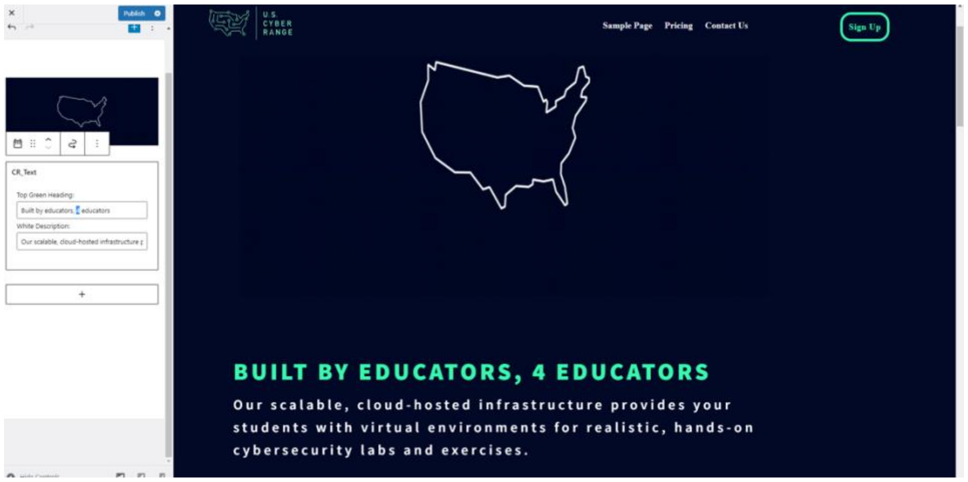
45 Used by thousands of students and educators in 45 states

45 Used by thousands of students and educators in 45 states

Figure 24 Customize Site View

By clicking the blue “Customize Your Site” button in the dashboard, the site will navigate you to an editable version of your website; an example is shown in Figure 24. On the left is a “Widgets” option which shows the different kinds of widgets that are being used on the site. There is an option to add or delete widgets in this menu, and the list of widgets presented includes the created custom widgets and the premade WordPress widgets.

Updating Widgets



CR_Text

Top Green Heading: Built by educators, 4 educators

White Description: Our scalable, cloud-hosted infrastructure p

BUILT BY EDUCATORS, 4 EDUCATORS

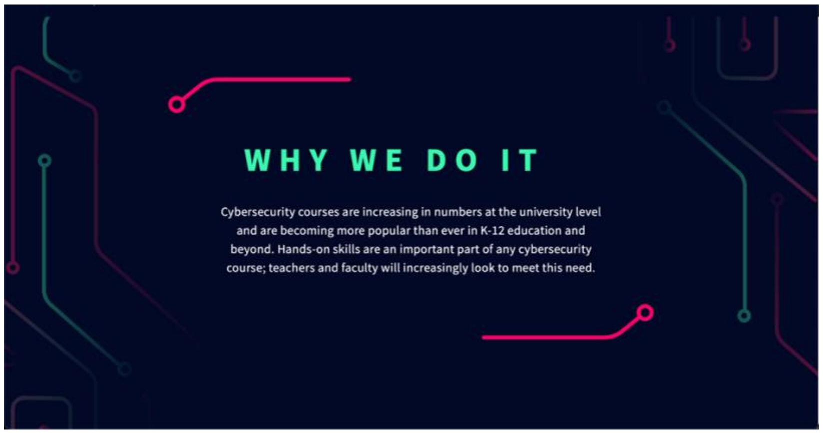
Our scalable, cloud-hosted infrastructure provides your students with virtual environments for realistic, hands-on cybersecurity labs and exercises.

Figure 25 Updating Widget

By clicking one of the widgets present on the website, you will be able to update the content that it contains. You can edit any field contained in a widget by changing the text in the provided textboxes. In order to save changes made to the content, the blue “publish” button at the top should be clicked. Figure 25 shows an example description widget being edited by the described process.

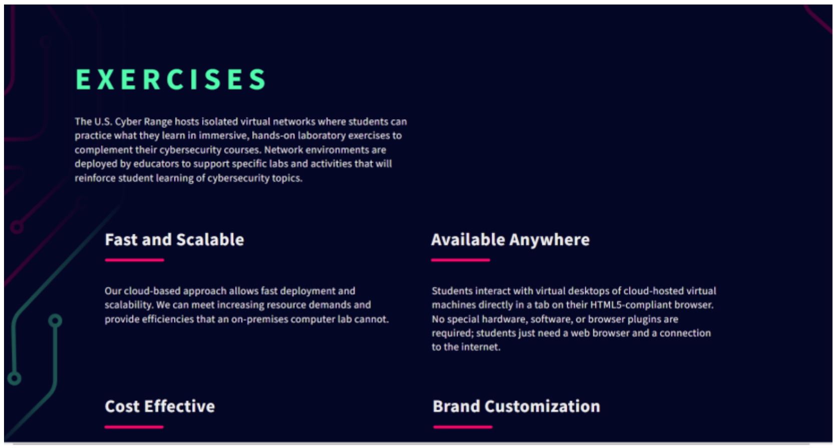
Work Completed

The header for all the pages was created. The about page has been almost finished. The only things that needs to be added are transitional animations for some of the items. Additionally, a footer was started. It is digitized but does not adhere to the visual design created by FourDesign. The home page and services page are started and contain a large amount of the styling/widgitizing needed. Currently nine custom widgets have been created and utilized in the implemented pages. For each of the mentioned pages, our team implemented the necessary CSS to match the design of the finalized pages in the Figma file. More significantly, our team created custom widgets for many sections of the website so that the client can easily modify or update the information that is presented. While the pages that we worked on are not fully widgetized, we have created documentation regarding all the steps we took for development. This ensures that our client and their future CS team can pick up where we left off and improve upon all the work that our team has completed.


WHY WE DO IT

Cybersecurity courses are increasing in numbers at the university level and are becoming more popular than ever in K-12 education and beyond. Hands-on skills are an important part of any cybersecurity course; teachers and faculty will increasingly look to meet this need.

Figure 26 Portion of About Page


EXERCISES

The U.S. Cyber Range hosts isolated virtual networks where students can practice what they learn in immersive, hands-on laboratory exercises to complement their cybersecurity courses. Network environments are deployed by educators to support specific labs and activities that will reinforce student learning of cybersecurity topics.

Fast and Scalable

Our cloud-based approach allows fast deployment and scalability. We can meet increasing resource demands and provide efficiencies that an on-premises computer lab cannot.

Available Anywhere

Students interact with virtual desktops of cloud-hosted virtual machines directly in a tab on their HTML5-compliant browser. No special hardware, software, or browser plugins are required; students just need a web browser and a connection to the internet.

Cost Effective

Brand Customization

Figure 27 Portion of the Services Page

Figures 26 and 27 depict sections of the About page and Services page, respectively. As shown in these figures, our team made sure that textual information across the website was fully customizable to ensure client maintainability. Our team also ensured that design elements were not lost during the implementation of the website. It was completed by referencing the CSS code present in the Figma [7] files and importing the necessary font family. It also required downloading assets and placing them accordingly throughout the website.

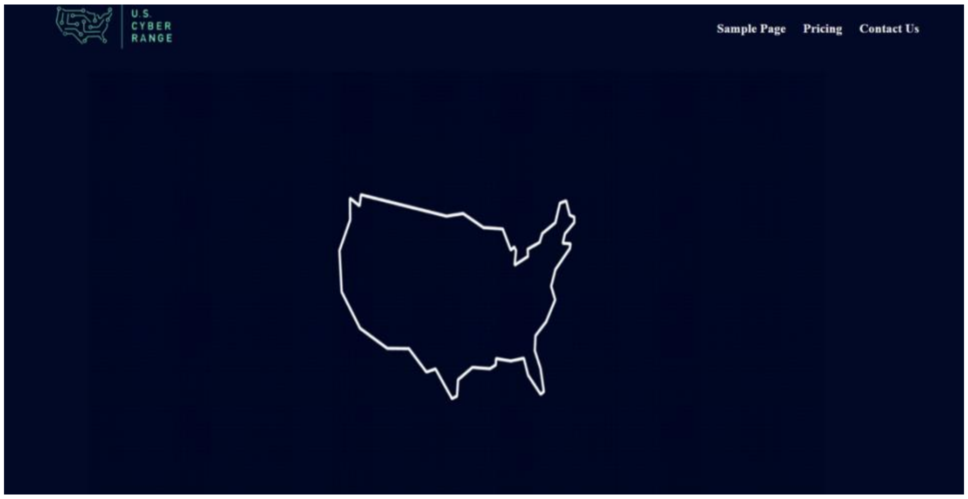


Figure 28 Portion of Home Page

The above figure showcases a section of the Home page. As shown in this figure, our team implemented image and animation widgets whenever possible. As seen in the figure above, our team also worked on creating the header and footer along with developing the four main pages. These pages were created with various custom widgets.
The first widget is the CR_About_Title. This widget creates a title out of white text with a green text section in the middle. A second widget, CR_About_Timeline, is a dynamic widget. It is used to create the timeline on the about page. A user can add and delete events to fill out the timeline. An event consists of a heading and a description. There is also a CR_Accented_Description widget which outputs a heading and description that is accented. A more general form of the CR_Accented_Description widget is the CR_Single_Description widget. This widget outputs a heading and description without the accent.

Another widget created is the CR_Text widget. This is a simple widget which outputs a plaintext field. The CR_Footer widget provides all the necessary editing for the footer on all the pages. Additionally, a CR_Feature_Images widget exists. This widget allows the feature section of the about page to be editable. The next widget, CR_Services_Title, provides the necessary fields to edit the title on the Services page. The final widget, CR_Double_Description, allows the user to output two headings, each with description text underneath.

Lessons Learned

Difficulties

The project had a slow start. A couple of weeks of coordination and communication between the client, FourDesign, and the CS team were required to get the project approval. Multiple lessons about project management were learned during this period such as the Computer Science team having to learn how to work in a multilevel team environment. Because of this workflow pipeline between the US Cyber Range, the FourDesign team, and the Computer Science team, the flow of communication was an aspect of this project that needed to be mastered in order to progress effectively with the project through the CS4624 class perspective. At times, remote communication between the Computer Science team and the FourDesign team was unclear or delayed on both ends, which contributed to the project's ambiguity. Furthermore, no webpage code implementation could be done until the FourDesign team was granted approval of their wireframes from their client, the US Cyber Range, which delayed much of the technical work until well into the latter half of the semester.

Another obstacle was the WordPress CMS system. Before the project started, the CS team had little to no prior experience with WordPress, namely creating custom widgets. The designs that FourDesign introduced were arduous to replicate in WordPress. Additionally, Virginia Tech was unable to grant the client access to a hosting server on the campus to host a professional version of WordPress for centralized source control for the Computer Science team to use. Consequently, this necessitated the CS team having to use workarounds such as Docker and XAMPP to obtain and work with the free version of WordPress.

Takeaways and Solutions

In hindsight, the best takeaway is the maintain constant and clear communication amongst all parties involved in a project. The project initially started off with communication with the client via email, which at times was easy to miss or forget about in terms of notifications. This resulted in less consistent as well as choppy communication between the Computer Science team and FourDesign. In order to solve this issue by having a platform for more accessible, quicker, and more organized communication, the FourDesign team added the Computer Science team to their Slack channel, which was convenient since this was the platform the Computer Science team already used to communicate amongst themselves. Another source of ambiguity that was alleviated was the assets that the FourDesign team had in their planning stages when collaborating with the US Cyber Range team. The FourDesign team set up a Google Drive shared with the Computer Science team so it could access FourDesign’s individual assets that they created for the purpose of planning how to implement them into the source code of the website.

Although US Cyber Range approval of FourDesign's wireframes took some time, the Computer Science team became immersed in a different, yet insightful realm of project management, which was site planning. In collaboration with FourDesign, the Computer Science team offered their technical knowledge in website navigation and logistics to help create an intuitive sitemap. Furthermore, the inability of Virginia Tech to grant a server for WordPress enabled the team to learn about independent development spaces on their local machines though experimentation while gaining knowledge about programs such as Git source control, Docker, XAMPP, and MySQL.

Timeline

September

- Weeks 1 – 2
  - Contact with client and establish communication
  - Review available CS department resources
- Weeks 3 – 4
  - Design process with client begins
  - Research Advanced Custom Fields and WordPress
  - Provide input on proposed wireframes

October

- Weeks 1 – 2
  - Discuss available hosting resources with client
  - Create git repository and research hosting solutions
  - Research WordPress widgets
  - Provide input on proposed sitemap
- Weeks 3 – 4
  - Create Docker container for localhost deployment
  - Continue team input on design process
  - Begin site implementation

November

- Weeks 1 – 2
  - Switch from utilizing Docker to XAMPP (MySQL and Apache)
- Weeks 3 – 4
  - Continue site implementation

December

- Weeks 1 – 2
  - Implement remaining site pages
  - Create documentation
  - Final report and presentation submission
  - Meeting with client to present completed work

Future Work

As discussed, our team was not able to complete the full implementation of the new site within the timeframe. Of the site pages, the Pricing, Cloud CTF, and Teams pages have not yet been started. Future work would include building out these pages and necessary widgets within the custom theme.

Additional aspects of the website development that require future attention include search engine optimization (SEO) and web analytics. The inclusion of SEO into the new website is open to various implementation methods. However, U.S. Cyber Range approved only Matomo for use with web analytics. No progress has been made on implementation of SEO, other than concluding that is should just be an additional WordPress plugin to add.

The client wants to ensure that the accessibility of the site is up to certain industry standards. The website must comply with WCAG 2.1 Level AA accessibility standards; currently no effort has been committed to ensuring this standard. These standards should be ensured across all pages and interactions with the site, including retroactively to previously implemented pages. These standards will be assessed by a third party. Alongside implementing accessibility, future work should also include ensuring that the website supports varying screen resolutions and platforms.

Having encountered and gained experience with the design process over the course of the semester, we understand that the final website design will continue to evolve. Remaining work will include revisions to previously implemented or future pages as the design process continues between U.S. Cyber Range, FourDesign, and the C.S. Team.

Acknowledgements

Client - US Cyber Range

https://www.uscyberrange.org/contact-us
1880 Pratt Drive, Suite 2006, Blacksburg, Virginia 24060

FourDesign Team

School of Visual Arts at Virginia Tech, 203 Draper Rd., Blacksburg VA 24061

Jeff Joiner

Director of FourDesign
jjoiner@vt.edu

Dr. Edward Fox

Professor at Virginia Tech
fox@vt.edu

References

[1] Four Design, “About | FourDesign,” *FourDesign*, *2021*. [Online]. Available: http://fourdesign.co/about/. [Accessed: 08-Dec-2021].

[2] U.S. Cyber Range of Virginia Tech, “About: U.S. Cyber Range,” *uscyberrange.org*, 2019. [Online]. Available: https://www.uscyberrange.org/about. [Accessed: 08-Dec-2021].

[3] VMware, “Apache Friends,” Apache Friends RSS, 2021. [Online]. Available: https://www.apachefriends.org/index.html. [Accessed: 30-Nov-2021].

[4] Automattic Inc, “Main Page «WordPress Codex,” codex.wordpress.org, 2021. [Online]. Available: https://codex.wordpress.org/Main_Page. [Accessed: 14-Dec-2021].

[5] Oracle Corporation, “MySQL :: MySQL 8.0 Reference Manual,” Mysql.com, 2019. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/. [Accessed: 14-Dec-2021].

[6] The PHP Group, “PHP: PHP Manual - Manual,” www.php.net, 2021. [Online]. Available: https://www.php.net/manual/en/. [Accessed 14-Dec-2021].

[7] J. Banks, “Guide to developer handoff in Figma,” Figma, 2021. [Online]. Available: https://www.figma.com/best-practices/guide-to-developer-handoff/ [Accessed Dec. 15, 2021].

[8] FourDesign. 2021. Proposal to U.S. Cyber Range for Website Redesign. Virginia Tech, Blacksburg, VA.