

1. Code I have implemented

```
> Http > Livewire > Profile.php
class Profile extends Component
{
    public $user;


    protected $listeners = [
        'refresh' => '$refresh'
    ];

























    public function toggleFollow() {
        $currentUser = Auth::user();
        if ($currentUser->following->contains($this->user)) {
            $currentUser->following()->detach($this->user->id);
        } else {
            $currentUser->following()->attach($this->user->id);
        }
        $this->emit('refresh');
    }


    public function render()
    {
        $image = $this->user->image;
        $follower = $this->user->following->contains(Auth::user());
        $following = Auth::user()->following->contains($this->user);
        $profilePicture = "";
        if (empty($image)) {
            $profilePicture = '/images/defaultUser.png';
        } else {
            $profilePicture = "/" . $image->filename;
        }
        return view('livewire.profile', [
            'profilePicture' => $profilePicture,
            'follower' => $follower, 'following' => $following
        ]);
    }
}
```










This is the controller for a livewire component that shows a user's profile, containing their picture or the default picture if they don't have one, name, username and bio, along with a follow/unfollow button and a notification if they are following you.


2. Screenshots of git commit history







 Commits on Jan 7, 2023

Add following/unfollowing  RachelBirdy committed yesterday	 2b65d95 
Bug fix  RachelBirdy committed yesterday	 17b6073 
Bug fixes  RachelBirdy committed yesterday	 8a0fb09 
Add light theme and deletion  RachelBirdy committed yesterday	 3ad877d 
Add light theme  RachelBirdy committed yesterday	 9b056ff 
Add post editing  RachelBirdy committed yesterday	 82872ef 
Typo fix  RachelBirdy committed yesterday	 4e339b8 
Remove debug code  RachelBirdy committed yesterday	 388b56e 


 Commits on Jan 6, 2023

Modify roles  RachelBirdy committed 2 days ago	 142a2e8 
Add roles  RachelBirdy committed 2 days ago	 84a4096 
Add youtube service  RachelBirdy committed 2 days ago	 ef4db1a 

 Commits on Jan 5, 2023

Add ability to add new comments  RachelBirdy committed 3 days ago	 f3182e7 
Move comment view to one page  RachelBirdy committed 3 days ago	 d940fc3 


Bugfixes

 RachelBirdy committed 29 minutes ago

cf5f4aa



Formatting change

 RachelBirdy committed 3 hours ago

1795457




Make user index links

 RachelBirdy committed 3 hours ago

9c880c2




Add user index to menu

 RachelBirdy committed 3 hours ago

9adb3d9



Missed a bracket

 RachelBirdy committed 3 hours ago

49eafd3




Require auth on user index view

 RachelBirdy committed 3 hours ago

15c904f



Fix theme bug on profile view

 RachelBirdy committed 3 hours ago

2f3f5f3




Remove test route

 RachelBirdy committed 3 hours ago

4fdac39



Update login/register themes and pages

 RachelBirdy committed 3 hours ago

5f28eb4




Add redirect

 RachelBirdy committed 6 hours ago

3b43df1




Display pictures and video embeds with posts

 RachelBirdy committed 6 hours ago

95d0c0c



Typo fix

 RachelBirdy committed 7 hours ago

6181d19



Add videos and images to posts

 RachelBirdy committed 7 hours ago

a3f6778




Add youtube url validation

 RachelBirdy committed 7 hours ago

795d267



Settings redirect bugfix


 RachelBirdy committed 11 hours ago

dd83895



Commits on Jan 5, 2023

Fix image path storage


 RachelBirdy committed 3 days ago



c64a255



Fix image deletion in migration


 RachelBirdy committed 3 days ago



b9e4497



Layout updates


 RachelBirdy committed 3 days ago



9eca426



Star notifications work


 RachelBirdy committed 3 days ago



e5a9a3f



Add livewire stuff


 RachelBirdy committed 3 days ago



181aea0



Make image relations polymorphic


 RachelBirdy committed 3 days ago



08dcbb9



Add image file deletion on migration


 RachelBirdy committed 3 days ago



99465ed



Cause I'm a live! A live wiiiiire!

 RachelBirdy committed 3 days ago



1699201



Commits on Jan 4, 2023

Add user bio, various formatting changes

 RachelBirdy committed 4 days ago




acf9d7d



Commits on Jan 3, 2023

Add comment number counter


 RachelBirdy committed 5 days ago



94b554e



Add profile pictures


 RachelBirdy committed 5 days ago



8eb36a1



Add rudimentary theming

 RachelBirdy committed 5 days ago




3c75a23



Commits on Jan 2, 2023

 RachelBirdy committed 2 days ago

Add youtube service

 RachelBirdy committed 2 days ago




ef4db1a



Commits on Jan 5, 2023

Add ability to add new comments


 RachelBirdy committed 3 days ago



f3182e7



Move comment view to one page


 RachelBirdy committed 3 days ago



d940fc3



Move comments to livewire


 RachelBirdy committed 3 days ago



3508898



Move comments to livewire rendering


 RachelBirdy committed 3 days ago



49930fa



Move menu to livewire


 RachelBirdy committed 3 days ago



d47960a



Misc

 RachelBirdy committed 3 days ago



1afbacb



Add comment rendering livewire


 RachelBirdy committed 3 days ago



cfbeb29



Remove default files


 RachelBirdy committed 3 days ago



ed32879




































Move profile rendering to livewire

 RachelBirdy committed 3 days ago



25ae688



Commits on Jan 2, 2023		
Add profile pictures	 e6a5b2c	
 RachelBirdy committed last week		
Add following functionality	 fcc99b2	
 RachelBirdy committed last week		
Commits on Dec 29, 2022		
Added rudimentary profile navigation	 d2c6b29	
 RachelBirdy committed last week		
Added a silly little user listing test	 74eae4	
 RachelBirdy committed last week		
Undo most of the username stuff. It didnt work. :	 7df0721	
 RachelBirdy committed last week		
More username things	 26d1d8a	
 RachelBirdy committed last week		
add usernames	 574803b	
 RachelBirdy committed last week		
Commits on Dec 28, 2022		
Add logout functionality	 732beb5	
 RachelBirdy committed 2 weeks ago		
Add layout testing	 2df376e	
 RachelBirdy committed 2 weeks ago		
Commits on Nov 7, 2022		
Fixes	 b5d79eb	
 RachelBirdy committed on Nov 7, 2022		
Update factories	 d5bcef4	
 RachelBirdy committed on Nov 7, 2022		
Commits on Nov 6, 2022		

3. MVC

The model-view-controller pattern is a form of design that broadly splits applications and their UI into three components:

1. Models, which are used to store data on an object. As well as the attributes of the object being accessible via them, they also provide access to the relations that object has with another object.
2. Views, which are the pages displayed to the user. These typically contain minimal if any logic themselves, acting purely as HTML/PHP pages, with certain locations marked for data from a variable to be inserted and for additional sections to be added in.

```
@section('content')
<div>
    <livewire:create-post>
</div>
<div class="post">
    @foreach ($posts as $post)
        <livewire:post :post='$post'>
    @endforeach
</div>
<a href="/home?page={{ $page-1 }}">Previous page</a>
<a href="/home?page={{ $page+1 }}">Next page</a>
@endsection
```

This is a section being rendered. The posts to be displayed have already been retrieved, processed and ordered before being passed into the view by the controller.

3. Controllers, which perform any logic operations and perform actions such as creating, updating retrieving or destroying models.

```
/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return Image->id
 */
public function storeProfilePicture(Request $request)
{
    $path = $request->file('image')->store('images');
    $image = new Image;
    $image->filename = $path;
    $image->imageable_id = $request->user()->id;
    $image->imageable_type = "App\Models\User";
    $image->save();
    $id = $image->id;
    return $id;
}
```

This is a typical function in a controller. When accessed via a post request defined in the routes file, it saves an uploaded picture, creates an Image object, assigns the path of the saved image to the Image object, and saves the object to the database.

In an MVC system, the user sends a request to a particular function on a controller. The controller, interacts with and collates data from any models that it needs, or updates or performs any other operations on a model depending on the request. The model then returns a view to the user, with any data it's collected passed into it and slotted into the marked locations.

4. WCAG2 reflection

The site meets a very limited number of WCAG2 guidelines

1 Perceivable

Text alternatives

There are no text alternatives to images and no ability to add it to user uploaded images.

Time based media

The only time-based media on the side is embedded youtube videos, and youtube provides automated and manual captioning which carries over to embedded videos and streams.

Adaptability

Labelling though present in places is inconsistent. The tables used to present the app should linearize correctly, data is presented in a meaningful sequence, but no explicit indicators of sequence or ordering have been provided.

Instructions for operating content do not make any reference to the visual layout/sensory characteristics of the page (i.e. there is no “Type in the box below” or “Click the red button” that would become meaningless if the page were interpreted differently)

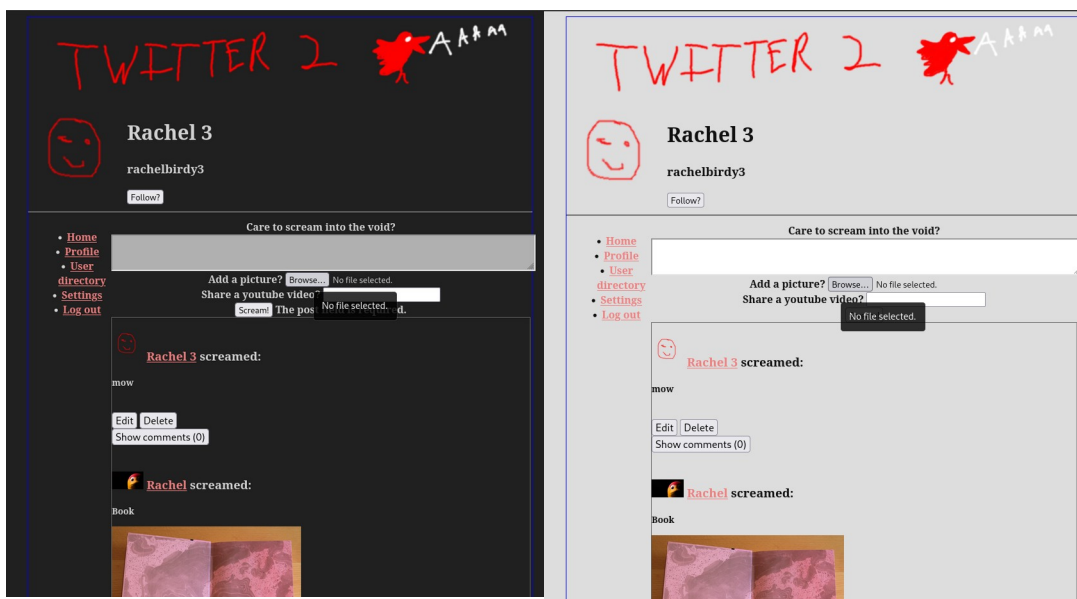
Content is constrained in it's presentation to a narrow, portrait view.

Distinguishably

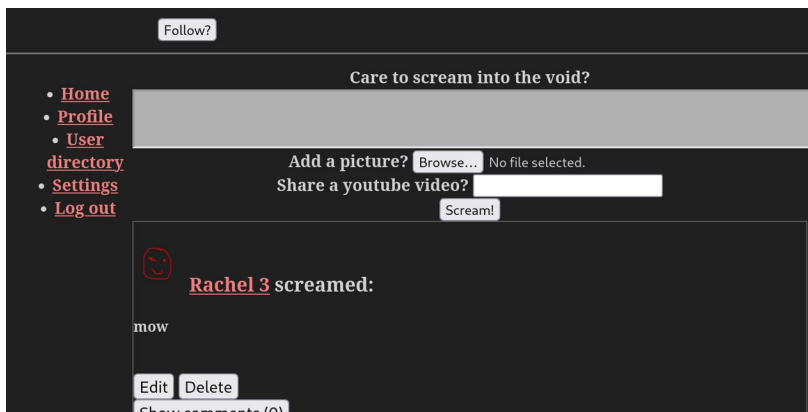
Colour is not used to convey information, prompt actions or distinguish visual elements

There is no audio which automatically plays.

The site layout is high contrast, contrasting white text against dark backgrounds or vice versa depending on the theme in use. It does not, however, have an explicit high contrast mode that ensures that all elements meet the 4.5:1 and 7:1 contrast requirements of the guidelines.



Resizing text has no negative impact on the site's functionality, as shown in this screenshot of the site at 200% magnification in browser.



All text on the site outside of the logo is conveyed using text rather than images of text

Foreground and background colours can be selected in a limited capacity by the user – there are multiple themes available that change this, but they cannot be selected as a free choice.

2 Operable

Keyboard accessible

All functionality of the site can be accessed using only a keyboard, there are no traps which pull and keep focus and require non-standard ways to navigate away from, and there are no character key shortcuts on the site

Enough time

There are no timing dependant elements to the site. There are no auto-updating elements, interruptions, or any other elements to the frontend which are time dependant. However, if a user's authentication expires and they have to reauthenticate, any data stored in forms that was not saved will be lost. There is no warning that this will happen.

Seizures and physical reactions

The site does not contain any flashing content, and user interactions do not trigger any animations

Navigable

Though repeated content between pages is minimal, there is no functionality to skip it.

Page titles are present but inconsistent

The order that elements are focussed on when tabbing through is logical, with different sections of the site grouped together.

All links purposes are clearly signposted by the text they are assigned to.

There are not multiple ways to get to some pages on the site. For instance, the only way to get to the settings page, is by clicking the settings link in the menu on the left, there is no alternate route.

However, user profiles and content can be navigated to by clicking usernames anywhere they appear, in posts, comments, or in the user directory.

Headings and labels are inconsistent and unreliable.

The keyboard focus indicator is visible on almost all elements when tabbing through the site. The only elements it is not visible on are embedded youtube videos.

There is little information about the users location within a set of webpages available. Some pages, but not all, have titles which indicate this.

Input modalities

There are no gesture based controls in the interface.

All pointer functions are completed on the mouse up event, the down event is not used.

Input mechanisms are not limited by the applications

3 Understandable

Readable

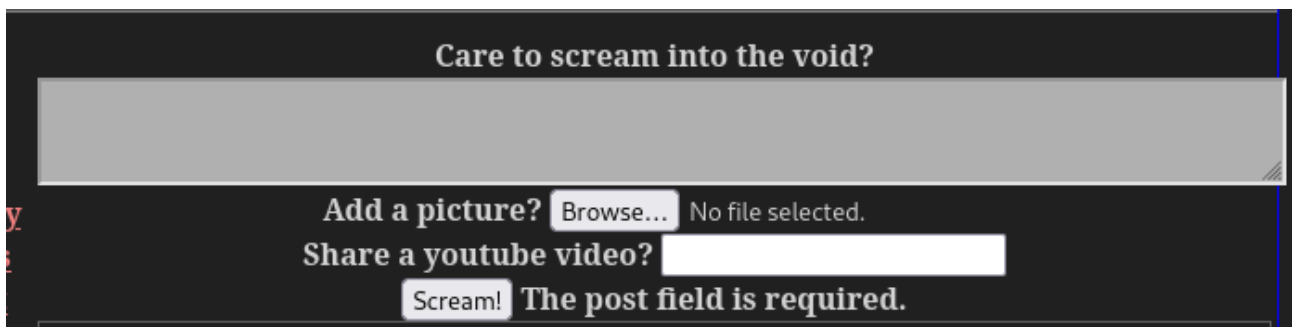
Because the site relies on user generated content, there is no way to assess the reading ability required for the content, nor to programmatically determine the language of the content. There is also no ability to identify specific definitions or pronunciations of words in context.

Predictable

An element receiving focus never changes context. User input does not change context until the user actively submits the form using the submit button or enter key. Navigation is consistent across the site, with the left hand menu always being present. Changes of context are only ever user initiated.

Input assistance

Validation is present and input errors are presented to the user.

A screenshot of a web form with a dark background. At the top, the text "Care to scream into the void?" is displayed in a light color. Below this is a large, empty rectangular input field. Underneath the input field, there are two lines of text: "Add a picture?" followed by a "Browse..." button and the text "No file selected.", and "Share a youtube video?" followed by an empty text input field. At the bottom of the form, there is a "Scream!" button and a message "The post field is required." in a light color.

Labels and instructions for user input are lacking on the post form, and the comment box has no instruction on how to submit a comment. There is no suggestion of corrections for errors, and there is no context sensitive help.

4 Robust

Compatibility

Not all elements have complete start and end tags, the table on the main view is left open without an end tag. Element ID's are unique and anywhere a shared identifier is needed for css styling a class is used rather than an ID.

All form elements and links are standard HTML controls

The site does not present status messages to the user except for in limited situations.

5. Service container

I designed a service which communicates with google's api to retrieve the embedding HTML for a youtube video based on a provided video ID. It is initially constructed with an API key from google, and contains a function that when called with a video id makes a call to the api for data on that video, parses the response and returns a string of HTML for embedding that specific video that can be inserted into a view.

```
<?php

namespace app\Http;

class Youtube {
    private $key;
    public function __construct($key) {
        $this->key = $key;
    }

    public function getVideo($watchId) {
        $url = "https://www.googleapis.com/youtube/v3/videos?id=".$watchId."&key=".$this->key.
            "&part=player";
        $embed = json_decode(file_get_contents($url), true);
        return $embed['items'][0]['player']['embedHtml'];
    }
}
```

The service container contains all of the applications bindings, which allow functions and objects to be accessed universally without being explicitly passed in to methods from the methods calling them.

```
app()->singleton('App\Http\Youtube', function($app) {
    return new Youtube('AIzaSyAN-3o7SaqfEckM4W-TsRrA2wCiM0rLaGI');
});

app()->make('App\Http\Youtube');
```

Binding “App\Http\Youtube” to a function that returns an instance of the class means that if this class is referenced from anywhere in the application where an explicit instance of the class has not already been passed in, this function will be called to return one. Binding it as a singleton specifically means that this function will only ever be run once, and any subsequent calls will return the same object generated by that first run.

```
36
37     public function render(Youtube $y)
38     {
```

An instance of the class can then be called inside functions without being passed in. Every time this render function is called, the function that the Youtube class is bound to will be called to get an instance of it.

This is useful for communicating with external API's because it lets all of the configuration for those API's be done in one place. Rather than having API keys scattered throughout various controllers it can all be set in one place, and the behaviour of the service across the application can be changed in one place. For instance, if I decided I don't like youtube anymore and want to use tiktok videos instead, I can change the code in the service and everywhere else it'll just work.