WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Individual Project 1

Front-end Web Development with a Professional Profile Website on github.io cloud service

Student

Name: Emily Turcotte

Email: turcotea@mail.uc.edu

Figure 1: Emily's Headshot

Overview and Requirements

Site URL: https://eturcotte270.github.io/

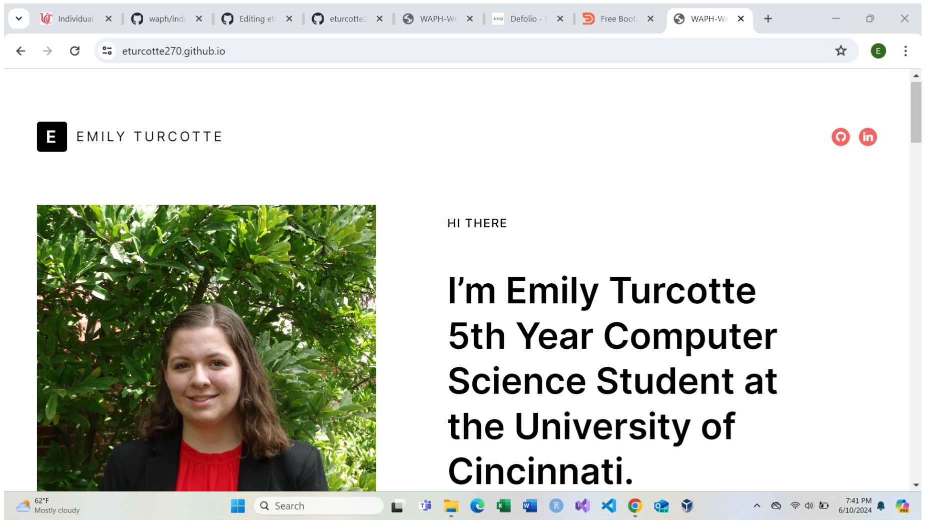
In this project, you will expand front-end web development skills by developing a Professional Profile Website and deploying it on `github.io` cloud service. This project has general, non-technical, and technical requirements with grade distributions as follows.

Outcomes I learned: + Deploying a page on GitHub cloud + Linking to other HTML pages + Using a Bootstrap template + Using JavaScript jQuery and React + Integrating APIs both that update and use graphics + Using Cookies

General requirements:

- Create and deploy a personal website on GitHub cloud (github.io) as a professional profile with your resume, including your name, headshot, contact information, background, e.g., education, your experiences and skills.

I completed this task by my main page being my profile and then a link to a page that contains Individual Project 1 as I intended to use this long term. The below screenshot shows my main page:



HI THERE

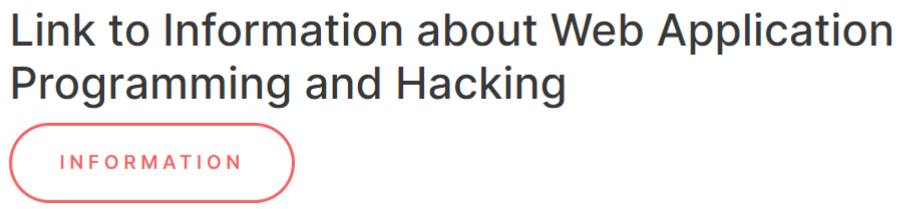
I'm Emily Turcotte 5th Year Computer Science Student at the University of Cincinnati.

Figure 2: Professional Profile

- Create a link to a new HTML page to introduce this “Web Application Programming and Hacking” course and related hands-on projects

I completed this task by creating a button to link to waph.html, and I used the following code:

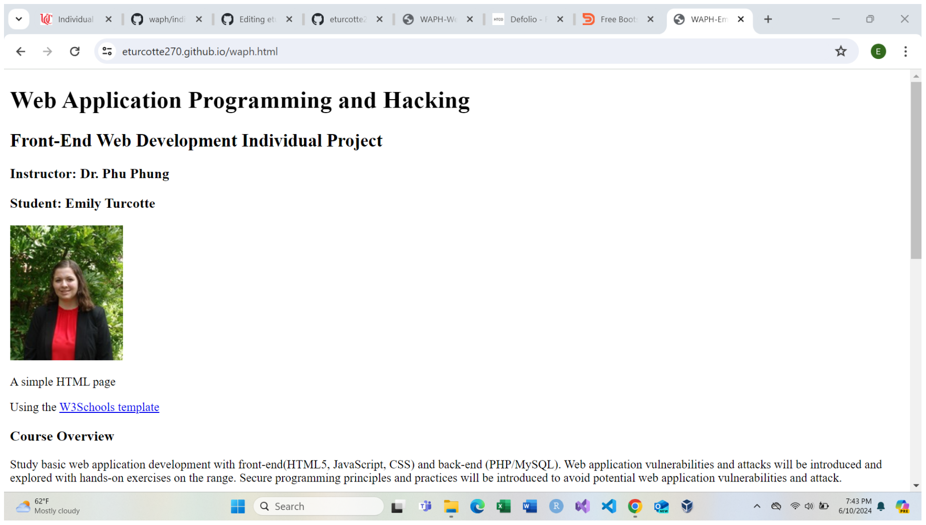
```
<a href="waph.html" class="ds-button">Information</a>
```



Link to Information about Web Application Programming and Hacking

INFORMATION

Figure 3: Link to waph.html



eturcotte270.github.io/waph.html

Web Application Programming and Hacking

Front-End Web Development Individual Project

Instructor: Dr. Phu Phung

Student: Emily Turcotte

A simple HTML page

Using the W3Schools template

Course Overview

Study basic web application development with front-end(HTML5, JavaScript, CSS) and back-end (PHP/MySQL). Web application vulnerabilities and attacks will be introduced and explored with hands-on exercises on the range. Secure programming principles and practices will be introduced to avoid potential web application vulnerabilities and attack.

Figure 4: waph.html page

Non-technical requirements

- Use an open-source CSS template or framework such as Bootstrap https://www.designstub.com/product/defolio-bootstrap-5-html-resume-template/

I used the template above and added my code in index.html, works-details.html, and waph.html.

- Include a page tracker, https://flagcounter.com/.

To include the page tracker for location of visitors I added the following code:

```
<a href="http://s11.flagcounter.com/more/g9y"><img src="https://s11.flagcounter.com/count2/g
```

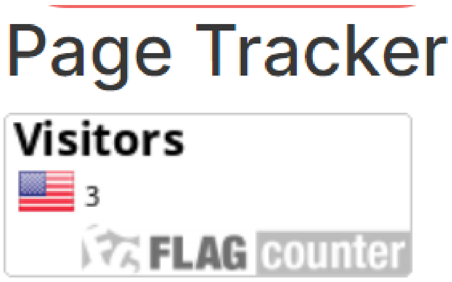


Figure 5: Page Tracker - Visitor Location

Technical requirements

Basic JavaScript code Use jQuery and one more open-source JavaScript framework/library to implement JavaScript code introduced in Lab 2, including a digital clock, an analog clock, show/hide your email, and one more functionality of your choice.

To add jQuery I added the following code:

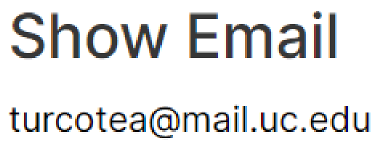
```
<script src="https://code.jquery.com/jquery-3.7.1.min.js"
    integrity="sha256-/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo="
    crossorigin="anonymous"></script>
```

To add React I added the following code:

```
<script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></scri
```

Show/hide email To show/hide my email I added the following HTML and JavaScript code:

```
<div id="email" onclick="showhideEmail()">Show my email</div>

    var shown = false;
    function showhideEmail(){
        if (shown){
            document.getElementById('email').innerHTML = "Show my email";
            shown = false;
        } else {
            var myemail = "<a href='mailto:turcotea" + "@" + "mail.uc.edu'>turcotea" + "@" -
            document.getElementById('email').innerHTML = myemail;
            shown = true;
        }
    }
```


Show Email

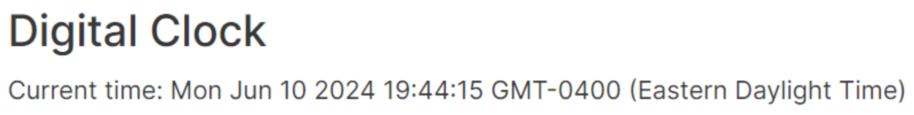
turcotea@mail.uc.edu

Figure 6: Show/Hide Email

Digital Clock To add a digital clock I added the following HTML and JavaScript code:

```
<div id="digit-clock"></div>
```

```
    var shown = false;
    function displayTime() {
        document.getElementById('digit-clock').innerHTML = encodeInput("Current time: " + ne
    }

    setInterval(displayTime, 500);

    function encodeInput (input) {
        const encoded = document.createElement('div');
        encoded.innerText = input;
        return encoded.innerHTML;
    }
```



Digital Clock

Current time: Mon Jun 10 2024 19:44:15 GMT-0400 (Eastern Daylight Time)

Figure 7: Digital Clock

Analog Clock To add an analog clock I added the following HTML and JavaScript code:

```
<div id="digit-clock"></div>
```

```
    var canvas = document.getElementById("analog-clock");
    var ctx = canvas.getContext("2d");
    var radius = canvas.height / 2;
    ctx.translate(radius, radius);
    radius = radius * 0.90;
    setInterval(drawClock, 1000);

    function drawClock() {
        drawFace(ctx, radius);
        drawNumbers(ctx, radius);
        drawTime(ctx, radius);
    }
```

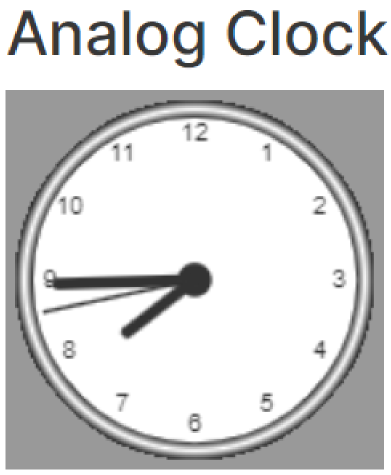
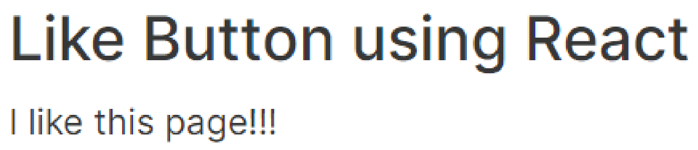


Figure 8: Analog Clock

Like Button using React To add the like button using React I added the following HTML and JavaScript code:

```html
<div id="like_button_container"></div>
<script src="assets/js/like_button.js"></script>
```

```javascript
'use strict';
const e = React.createElement;
class myButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { isliked: false };
  }

  render() {
    if (this.state.isliked) {
      return 'I like this page!!!';
    }

    return e(
      'button',
      { onClick: () => this.setState({ isliked: true }) },
      'Like Button'
    );
  }
}
const domContainer = document.querySelector('#like_button_container');
ReactDOM.render(e(myButton), domContainer);
```


Like Button using React

I like this page!!!

Figure 9: Like Button Result

Two public Web APIs integration

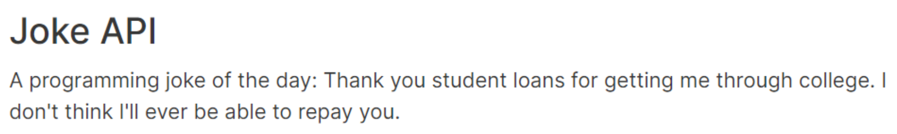
Joke API

1. Integrate the jokeAPI (https://v2.jokeapi.dev/joke/Any) with `Any` category of joke to display a new joke in your page every 1 minute.

To add the jokeAPI and have it update every minute I added teh following HTML and JavaScript code:

```html
<div id="joke-response"></div>
```

```javascript
function apiCall() {
    var baseURL = "https://v2.jokeapi.dev/joke/Any";


    var xhr = new XMLHttpRequest();
    xhr.open("GET", baseURL);

    xhr.onreadystatechange = function() {
        if(xhr.readyState == 4 && xhr.status < 300)
        {
            var randomJoke = JSON.parse(xhr.responseText);

            if(randomJoke.type == "single")
            {
                $("#joke-response").html("A programming joke of the day: " + randomJoke.joke
            }
            else
            {
                $("#joke-response").html("A programming joke of the day: " + randomJoke.setu
            }
        }
        else if(xhr.readyState == 4)
        {
            $("#joke-response").html("Error while requesting joke.\n\nStatus code: " + xhr.
        }
    };

    xhr.send();
}

apiCall();
setInterval(apiCall, 60000);
```


Joke API

A programming joke of the day: Thank you student loans for getting me through college. I don't think I'll ever be able to repay you.

Figure 10: JokeAPI updates every minute

Dog API

2. Integrate a public API with graphics and display that graphic/image in your page.
   https://dog.ceo/api/breeds/image/random

To add the graphical API that displays images of dogs I added the following HTML and JavaScript Code:

```html
<div id="dog-response"></div>
```

```javascript
$.get("https://dog.ceo/api/breeds/image/random",
    function(result) {
        if (result.length == 0) return;
        const imageElement = document.createElement("img");
        imageElement.src = result.message;
        const container = document.getElementById("dog-response");
        container.appendChild(imageElement);
})
```

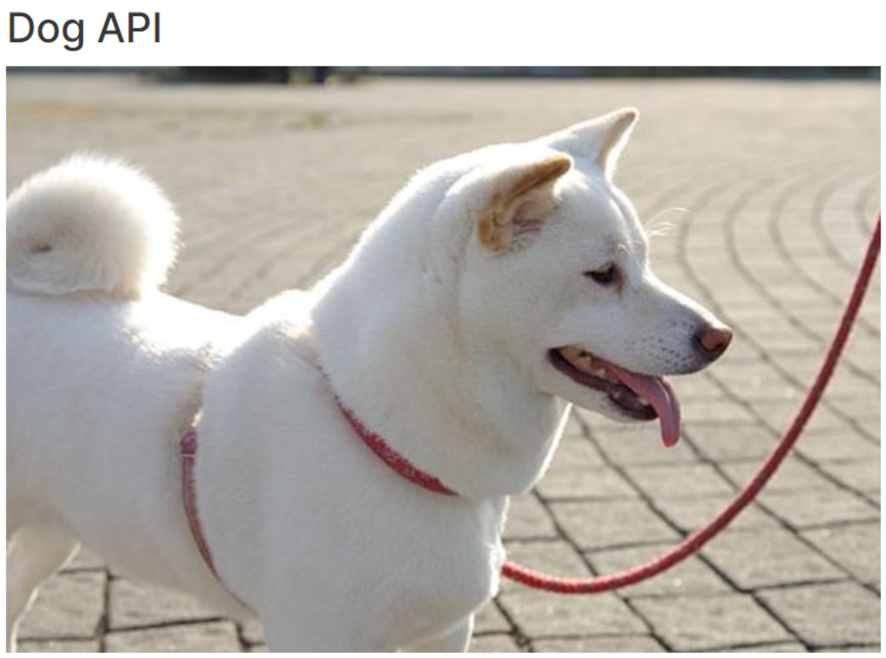


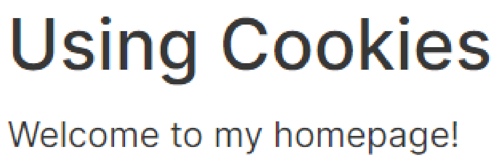
Figure 11: Graphical API - Dogs

JavaScript Cookies Use JavaScript cookies to remember the client: If first-time visit, display the message “Welcome to my homepage for the first time!”; otherwise, display the message “Welcome back! Your last visit was ”.

To add the display message using JavaScript Cookies I added the following HTML and JavaScript code:

```
<div id="cookies-response"></div>
```

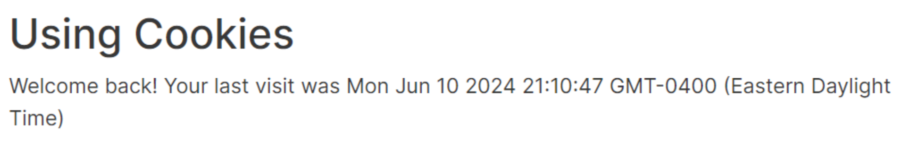
```
if (document.cookie.indexOf("visit") <0 ){
    $("#cookies-response").html("Welcome to my homepage!");
    document.cookie = "visit=" + new Date();
} else {
    $("#cookies-response").html("Welcome back! Your last visit was " + document.cookie.subst
    document.cookie = "visit=" + new Date();
}
```



Using Cookies

Welcome to my homepage!

Figure 12: First Visit



Using Cookies

Welcome back! Your last visit was Mon Jun 10 2024 21:10:47 GMT-0400 (Eastern Daylight Time)

Figure 13: Return Visit