Code Review Documentation

Jonathan Vargas - 200389468

Capstone Project - F.L.O.A.T.

April 9, 2022

Web Development Code Overview

My contribution to F.L.O.A.T. (Facilitating Level Objectives Assessment Technology) was to set up our backend server to process the data from Carter's algorithm and sensor readings from our Raspberry Pi. With this processed data I was in charge of creating a simple web client for our customer (government officials) to display the data so the customer is informed of the water parameters and make their next action plan to care for the water body.

Table of Contents

1. Design

1.1. Design Purpose

The purpose of this project is to create convenience for surveying and a simple minimalistic web client was the best way to translate our goal. Throughout the website the user experience is kept minimalistic with a single feature for a user to delve deeper into complexity.

1.2. Design patterns

Majority of the client side of the project was designed following the MVC (Model View Controller) design pattern with a slight use of the decorator pattern to display our map.

1.3. MVC Design Pattern

With the interface buttons acting as the user's controller I utilized php to manipulate the view (webpage) that the user sees. With the php code our website is able to display different data using only a few editable web pages. This design pattern creates ease of development for the programmer and server processing times.

1.4. Decorator Pattern

This design pattern is only used to create the map pins that are displayed in a small section of the website. This is used partly due the execution of the design patterns and the nature of the third party library I was utilizing. To display a warning on the map a marker object has to be formed and to account for multiple nested warnings the decorator design pattern was used to continuously apply different warnings.

2. Functionality

Based on a few internal tests against the outlined User Story Map outlined at fall 2021 it is able to perform the tasks set out reliably. I was regrettably unable to perform robust user tests. I have also found some aesthetic dysfunctionalities as seen in figure 1. System provides accurate feedback when submitting a form, however feedback is not consistent to log in screen. Data parsing handles smoothly with well formatted files. The system also handles missing data elegantly with only small portions of the view being unrendered.

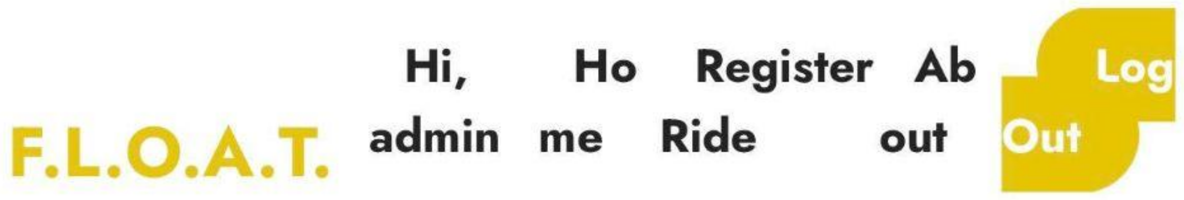
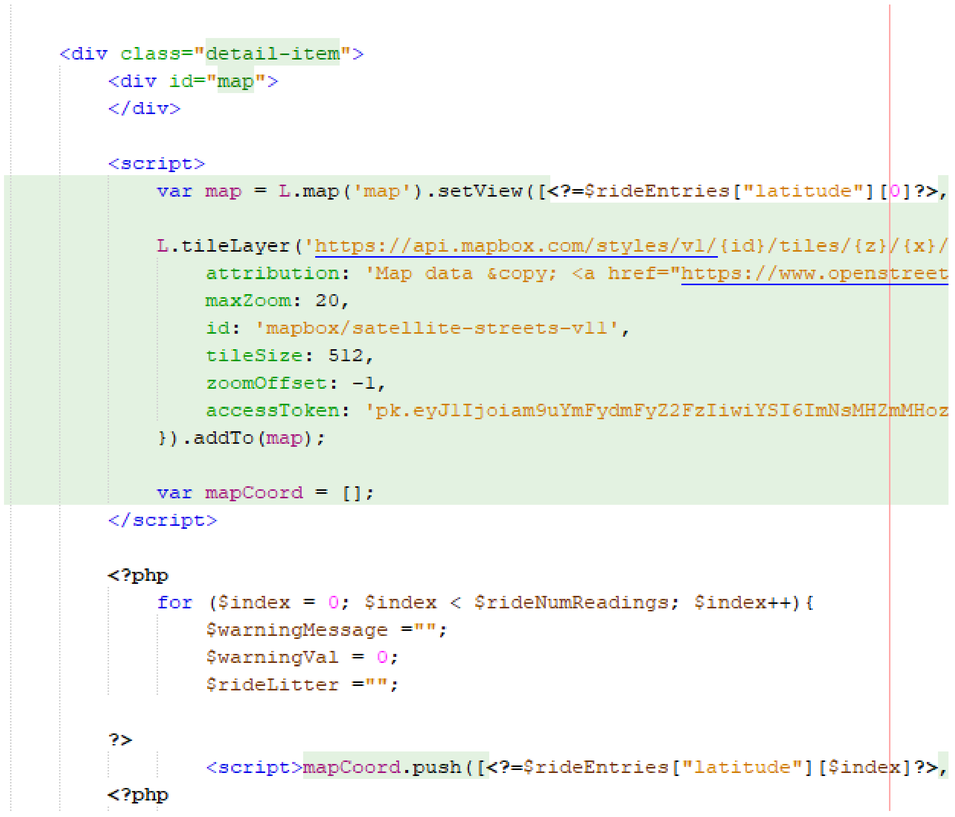


Figure 1. Navigation bar when in mobile view

3. Complexity

This project demanded lots of unique classes and functions that were difficult to separate into callable functions and classes. Majority of the code is written in document snippets to cascade different html and javascript sections of the web page; an example is shown in Figure 2. These code sections are due to my limited knowledge of dom manipulation with javascript. With more practice I am able to reduce the complexity of the code by creating proper class structure in php and javascript.



```
<div class="detail-item">
    <div id="map">
    </div>

    <script>
        var map = L.map('map').setView([<?=$rideEntries["latitude"][0]?>,

        L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/
            attribution: 'Map data &copy; <a href="https://www.openstreet
            maxZoom: 20,
            id: 'mapbox/satellite-streets-v11',
            tileSize: 512,
            zoomOffset: -1,
            accessToken: 'pk.eyJ1Ijoiam9uYmFydmFyZ2FzIiwiYSI6ImNsMHZmMHoz
        }).addTo(map);

        var mapCoord = [];
    </script>

    <?php
        for ($index = 0; $index < $rideNumReadings; $index++){
            $warningMessage ="";
            $warningVal = 0;
            $rideLitter ="";

    ?>
            <script>mapCoord.push([<?=$rideEntries["latitude"][$index]?>,
    <?php
```

Figure 2. Code snippet of inline code sections

With the use of these interlocking code sections I was able to keep certain aspects of the code DRY by adding a few include php statements which inject HTML code into every page that needed core aspects such as the navigation bar and the necessary links needed for styling.

My way of writing the php code may be un optimal but I believe it keeps it provides a certain flow to my code which allows me to read my code like a book and use minimal to no comments at all. This is due to grouping like lines of code together to create ‘chapters’ in the code.

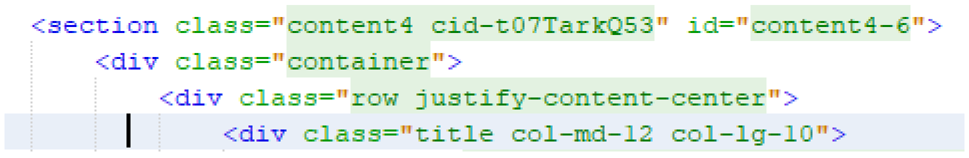
3.1. Use of third party software

At the start of this project I was against integration of third party software and only used open source software. However while designing the system I found that use of a couple would save some time, but would add unnecessary complexities to the system. For web page layout I used the software Mobirise and to display the map I used a recommended javascript library called leaflet. Developing with these software forced some interesting interactions. Leaflet forced me to design the map display with the decorator pattern due to the nature of how the library was built. Mobirise conveniently layed out the HTML elements however created complexities with hard to understand naming conventions.

4. Naming

Throughout the code I kept my values and page names simple, consistent, and descriptive. All values are in camel case with minimal abbreviations. Some values like values extracted from a ride csv file are in Hungarian Notation to separate them from other internal values. Pages are of higher importance which is why I opted with pascal case for naming files.

As mentioned earlier the use of mobirise created difficult to read class names when it comes to styling. Figure 3 shows the non descriptive class names generated.



```
<section class="content4 cid-t07TarkQ53" id="content4-6">
    <div class="container">
        <div class="row justify-content-center">
            <div class="title col-md-12 col-lg-10">
```

Figure 3. Style Class names

Conclusion

Although I had previous exposure with HTML and php this project taught me that I still have more to learn within the realm of web development. Even with my limited knowledge I believe I created readable semi-clean code. The code reads like a book with neat grouping and descriptive developer naming conventions. The code is not clean in areas such as unnecessary complexity and lack of OOP structure. Introductions of software generated code introduced hard to read class names.