

Lab 5 - AJAX Bus Tracker

60 Possible Points

1/25/2022

Attempt 1

**IN PROGRESS**

Next Up: Submit Assignment



Add Comment

Unlimited Attempts Allowed

1/11/2022

▼ Details

Introduction

In this assignment, you will write an application that uses AJAX to dynamically update a web page with data retrieved from a web server.

Objectives

By the end of the lab you will be able to:

- Develop JavaScript and Bootstrap development skills
- Enhance your HTML, CSS, and JavaScript development skills
- Use JavaScript fetch to send and received data from a web server using an AJAX request.

Background

The Milwaukee County Transit System (MCTS) maintains a website where you can obtain information about busses currently running on various routes.

You can get a feel for the information available by looking at the **MCTS real-time online bus map** (<http://realtime.ridemcts.com/bustime/map/displaymap.jsp>): visit the page, click on Routes, select a route, and zoom in on the map.

This web server can also serve up raw data in JSON format that can be retrieved by anyone who knows how to access it. You can retrieve the data from any route - for instance, the Green Line (Bayshore to the airport) is labeled as "GRE".

~~NOTE: You will be using the MCTS API web service in this lab. Doing so requires a free developer~~

<https://msoe.instructure.com/courses/9417/modules/items/373694>

Assignment

<https://msoe.instructure.com/courses/9417/modules/items/373694>

- Create an account: <http://realtime.ridemcts.com/bustime/createAccount.jsp>
(<http://realtime.ridemcts.com/bustime/createAccount.jsp>)
- Once you create an account sign in and request an key by clicking on "My API" in the top navigation bar

NOTE: You must demonstrate that you have retrieved your key to your instructor by the end of lab.

Accessing the MCTS real-time data

You can access the MCTS data using the MSOE proxy by pointing your browser at the following:

<https://msoe-web-apps.appspot.com/BusInfo?key=XXXXXXXXXXXXXXXXX&rt=GRE>
(<https://msoe-web-apps.appspot.com/BusInfo?key=XXXXXXXXXXXXXXXXX&rt=GRE>)

Replace XXXXXXXXXXXXXXXX with your MCTS developer key

NOTE: You will have to be careful about how frequently you retrieve data from the MCTS website. More than one access per second may result in being locked out of the site. If locked out you will have to wait a couple hours for your account to be activated again.

Use this method to demonstrate that you have successfully created an MCTS account and key to your instructor.

References

The following is helpful reference:

- Leaflet API Documentation: <https://leafletjs.com/reference.html>
(<https://leafletjs.com/reference.html>)
- MCTS API Documentation:
http://realtime.ridemcts.com/bustime/apidoc/docs/DeveloperAPIGuide2_0.pdf
(http://realtime.ridemcts.com/bustime/apidoc/docs/DeveloperAPIGuide2_0.pdf)
- JavaScript Fetch API: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)

NOTE: for this assignment you will be using version 2 of the MCTS API. In order to access the document you will need to be logged in with your account.

<https://msoe.instructure.com/courses/9417/modules/items/373694>

<https://msoe.instructure.com/courses/9417/modules/items/373694>

For this lab it will be helpful to read through some tutorials on using Leaflet. The programming interface is not difficult, but takes some practice to get used to it. Thankfully there are plenty of tutorials available on the Leaflet web site (<https://leafletjs.com/examples.html>).

For this lab's exercise, complete the following tutorials on the Leaflet web site:

- Leaflet Quick Start Guide - <https://leafletjs.com/examples/quick-start/>
- Markers With Custom Icons - <https://leafletjs.com/examples/custom-icons/>

NOTE: The tutorials use MapBox (<https://www.mapbox.com/>) for the map data. For this lab we are going to use OpenStreetMap (<https://www.openstreetmap.org/>) since it can be used without needing an access key. To set up the map tile for the Leaflet Map using OpenStreetMap, use the following JavaScript:

```
const map = L.map(mapId).setView(position, 11);
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);
```

When you have completed the tutorials and feel comfortable using Leaflet with OpenStreetMap and custom markers, proceed to the lab project.

Lab Project

Work on the lab project is to be done individually. You are welcome to collaborate with class members, but the project must be your own work.

For the lab project, you will write a web page that regularly tracks the bus Information in a table format as well as on a Leaflet Map.

Leaflet (<https://leafletjs.com/>) is a free JavaScript library for displaying and interacting with maps. It lets you display a map at a particular location, add pinpoints, zoom in and out, etc. However, as a user of the library you are required to provide your own map image. For that we will use OpenStreetMap (<https://www.openstreetmap.org/>) which is free open source map data.

The behavior of your bus tracker should be as follows:

1. The user should select route via its route identifier

2. Initially the start button should be enabled and the "Stop" button should be disabled

(<https://msoe.instructure.com/courses/9417/modules/items/373694>)

(<https://msoe.instructure.com/cou>)

4. Map markers should have the bus number displayed below them

- HINT: <https://leafletjs.com/reference.html#tooltip>
(<https://leafletjs.com/reference.html#tooltip>)
- (<https://leafletjs.com/reference.html#tooltip>) See the parameter options for "permanent" which forces the tooltip to be "open" always
- Also, see the option for "offset" which allows you to move the tooltip relative to marker

5. New map markers should added as buses move every 5 seconds

NOTE: you do not need to remove old markers for buses while the tracker is running

6. When the mouse clicks on a marker, the bus's route appears in a popup.

HINT: <https://leafletjs.com/reference.html#popup> (<https://leafletjs.com/reference.html#popup>)

7. The map should stop updating (but continue to show existing markers) when the user clicks "Stop"

8. Selecting a new route and pressing "Start" causes new markers to be added for the new route

When everything is working, your tracker should look like this:



(<https://msoe.instructure.com/courses/9417/modules/items/373694>)

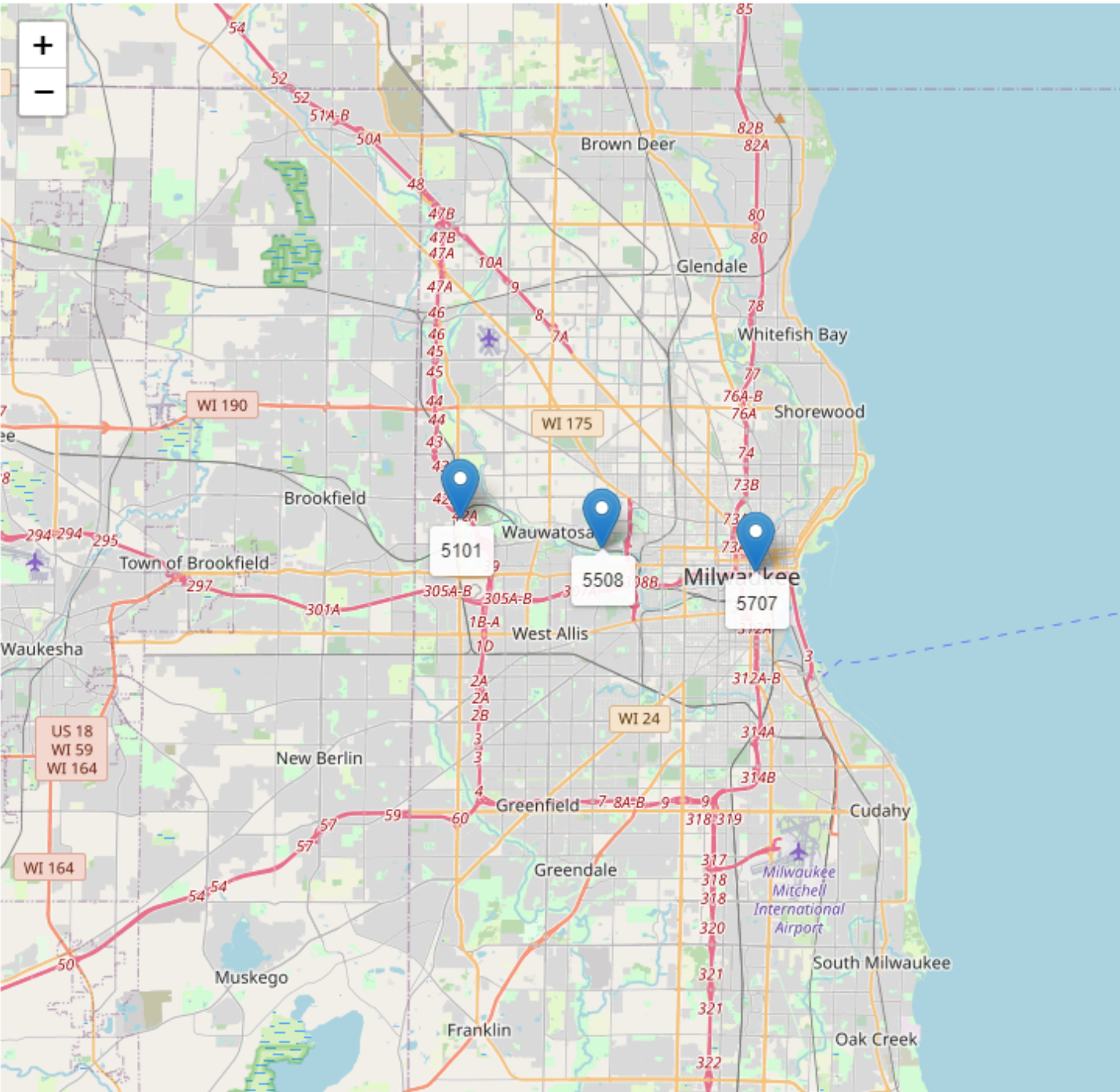


(<https://msoe.instructure.com/cou>)

Route:

Start

Stop



Bus	Route 31	latitude	longitude	speed
5707	Mayfair	43.035866666666664	-87.91668333333334	11
5101	Mayfair	43.052670328240644	-88.04740905761719	27
5508	Downtown-Intermodal	43.04344177246094	-87.98490142822266	26

- The "Start" button should be disabled
- The "Route" text input field should be disabled
- The "Stop" button should be enabled
- When not running:
 - The "Start" button should be enabled
 - The "Route" text input field should be enabled
 - The "Stop" button should be disabled

NOTE: If the user enters an invalid route or some other error message is returned from the MCTS API, display an appropriate error message to the user. Do **NOT** use an alert box for this. You do **NOT** need to clear the map markers or the table of rows when an error message occurs.

NOTE: If an error occurs at any time (either due to user input that is not valid or an error message from the server) stop updating **IMMEDIATELY**.

If the bus route information is successfully retrieved set an interval to update every 5 seconds (see **setInterval** (https://www.w3schools.com/jsref/met_win_setinterval.asp) documentation for help with this). At each update add new markers for the buses current location.

Recall that you can access the MCTS data using a URL like the following: <https://msoe-web-apps.appspot.com/BusInfo?key=XXXXXXXXXXXXXXXXX&rt=31> (<https://msoe-web-apps.appspot.com/BusInfo?key=XXXXXXXXXXXXXXXXX&rt=GRE>)

This request asks the server to supply a JSON collection of bus data for all buses currently on route 31. For details on the API supported by the MCTS web server, consult the **online documentation** (http://realtime.ridemcts.com/bustime/apidoc/docs/DeveloperAPIGuide2_0.pdf)

NOTE: You'll be using version 2.0 of the API. You must be logged into you free MCTS account in order to access this documentation.

The JSON response should look something like this (but it varies in real-time as buses move):

```
{
  "bustime-response": {
    "vehicle": [
      {
        "vid": "5316",
        "tmstp": "20210104 12:09",
        "lat": "43.04314809856993",
        "lon": "-87.92716506033233",
        "hdg": "359",
        "pid": 10674,
        "rt": "31",
        "des": "Innovation Drive (via State)",
        "pdist": 7044,
        "dly": false,
        "spd": 16,
        "tatrind": "34506333".
      }
    ]
  }
}
```



(<https://msoe.instructure.com/courses/9417/modules/items/373694>)



(<https://msoe.instructure.com/cou>)

```

      "tmstamp": "20210104 12:09",
      "lat": "43.042115347726",
      "lon": "-87.92717184339251",
      "hdg": "180",
      "pid": 10716,
      "rt": "31",
      "des": "Downtown/Intermodal",
      "pdist": 41893,
      "dly": false,
      "spd": 21,
      "tatripid": "34506309",
      "tablockid": "31 -304",
      "zone": ""
    },
    {
      "vid": "5213",
      "tmstamp": "20210104 12:08",
      "lat": "43.067054748535156",
      "lon": "-88.04571533203125",
      "hdg": "6",
      "pid": 10760,
      "rt": "31",
      "des": "Mayfair (via Vliet)",
      "pdist": 49233,
      "dly": false,
      "spd": 0,
      "tatripid": "34506321",
      "tablockid": "31 -301",
      "zone": ""
    },
    {
      "vid": "5212",
      "tmstamp": "20210104 12:09",
      "lat": "43.04003411370355",
      "lon": "-88.02081566887932",
      "hdg": "65",
      "pid": 10939,
      "rt": "31",
      "des": "Downtown/Intermodal",
      "pdist": 13593,
      "dly": false,
      "spd": 18,
      "tatripid": "34506300",
      "tablockid": "31 -302",
      "zone": ""
    }
  ]
}

```

You should also display each bus's information for the route in a table below the map. The table should display:

- The bus number - this is 'vid' in the MCTS results
- The bus destination - this is 'des' in the MCTS results
- The bus location (latitude and longitude) - these values are 'lat' and 'lon' in the MCTS results
- The bus speed - this is 'spd' in the MCTS results
- The bus traveling distance - this is 'pdist' in the MCTS results



(<https://msoe.instructure.com/courses/9417/modules/items/373694>)



(<https://msoe.instructure.com/cou>)

NOTE: The route number must be displayed in the header of the table

On each update the bus information for each bus should be updated in the table to reflect their current values. You can do this by directly updating the table or destroying and recreating the entire table.

Using the Fetch API

You will need to use the fetch JavaScript API to make your requests for the MCTS data. Remember that this API uses promises to perform the asynchronous request. You are welcome to use the .then/.catch interface or the async/await to manage the data. Recall that the response from a fetch represents the HTTP response as a whole. To get the JSON data from the body you will need to invoke the JSON API. This is also asynchronous and returns a promise.

Furthermore, recall that the fetch API only throws an error when there is a connection problem with the server. If it receives a response from the server, even if it is an error, then fetch will not reject the promise. You will have to manually check the response status to detect an error.

For example if you have a function called "updateMap" that updates the map based on JSON data and a function called "handleError" for handling errors, then using the .then/.catch you might write something like:

```
fetch(URL)
  .then((response) => {
    if(response.status !== 200) {
      throw new Error(String(response.status));
    }
    return response.json();
  })
  .then((responseJSON) => updateMap(responseJSON))
  .catch((error) => handleError(error));
```

Using async/await, it might look like:

```
try {
  const response = await fetch(URL);
  if(response.status !== 200) {
    throw new Error(String(response.status));
  } else {
    const responseJSON = await response.json();
    updateMap(responseJSON);
  }
} catch(error) {
  handleError(error);
}
```

However, these are just examples. Feel free to implement your fetch request as works best for you.



(<https://msoe.instructure.com/courses/9417/modules/items/373694>)



Assignment

(<https://msoe.instructure.com/cou>)

This application has been started for you. Remember to create your MCTS developer key before you get started.

Use the following files, and place your MCTS key in the BusTracker.js file.

- **BusTracker.html** (<https://msoe.instructure.com/courses/9417/files/1248505?wrap=1>) ↓
(https://msoe.instructure.com/courses/9417/files/1248505/download?download_frd=1)
- **BusTracker.js** (<https://msoe.instructure.com/courses/9417/files/1248506?wrap=1>) ↓
(https://msoe.instructure.com/courses/9417/files/1248506/download?download_frd=1)
- **BusTracker.css** (<https://msoe.instructure.com/courses/9417/files/1248507?wrap=1>) ↓
(https://msoe.instructure.com/courses/9417/files/1248507/download?download_frd=1)

The provided HTML is a starting point. Feel free to modify the structure and styling as you like. Just make sure your application meets the required functionality.

Possible changes you might want to make:

- Reorganize the user interface
- Move the map to a different place on the page
- Change the map size
- Alter other styling of the table, buttons, input field, etc.

Validation and Testing


Your application must be able to handle errors. There are several different errors that can be sent by the MCTS server. For example,


- No response from server (AJAX request fails due to a network failure, server timeout, or other server error)
Here, no JSON response will be received from the server - your AJAX error handler will be invoked
- Missing key or route in AJAX request
- Invalid key specified in AJAX request
- Invalid route specified in AJAX request

For the latter errors, different JSON responses will be received that you must distinguish from a normal response.

The proxy server has been purposely modified to allow you to force these errors to occur for testing purposes, as follows:

- If you supply a route parameter with value 1000, the server will simulate an internal error. The AJAX request will succeed but the JSON response will contain an error indication


(<https://msoe.instructure.com/courses/9417/modules/items/373694>)


Assignment
(<https://msoe.instructure.com/cou>)

- If you supply a route parameter with value 1002, the server will pretend that you didn't supply a key or route. The AJAX request will succeed, but the JSON response will contain an error indication.
- If you supply a route parameter with value 1003, the server will intercept the key you pass and substitute an invalid key. The AJAX request will succeed, but the JSON response will contain an error indication.

In all cases, you must present a meaningful error message to the user on the web page. Do **NOT** use an alert box to display an error message. Bootstrap has some components that might be useful for displaying error messages. For example, consider using a Bootstrap alert

(<https://getbootstrap.com/docs/4.0/components/alerts/>
(<https://getbootstrap.com/docs/4.0/components/alerts/>)).

Bad Route Number:



(<https://msoe.instructure.com/courses/9417/modules/items/373694>)

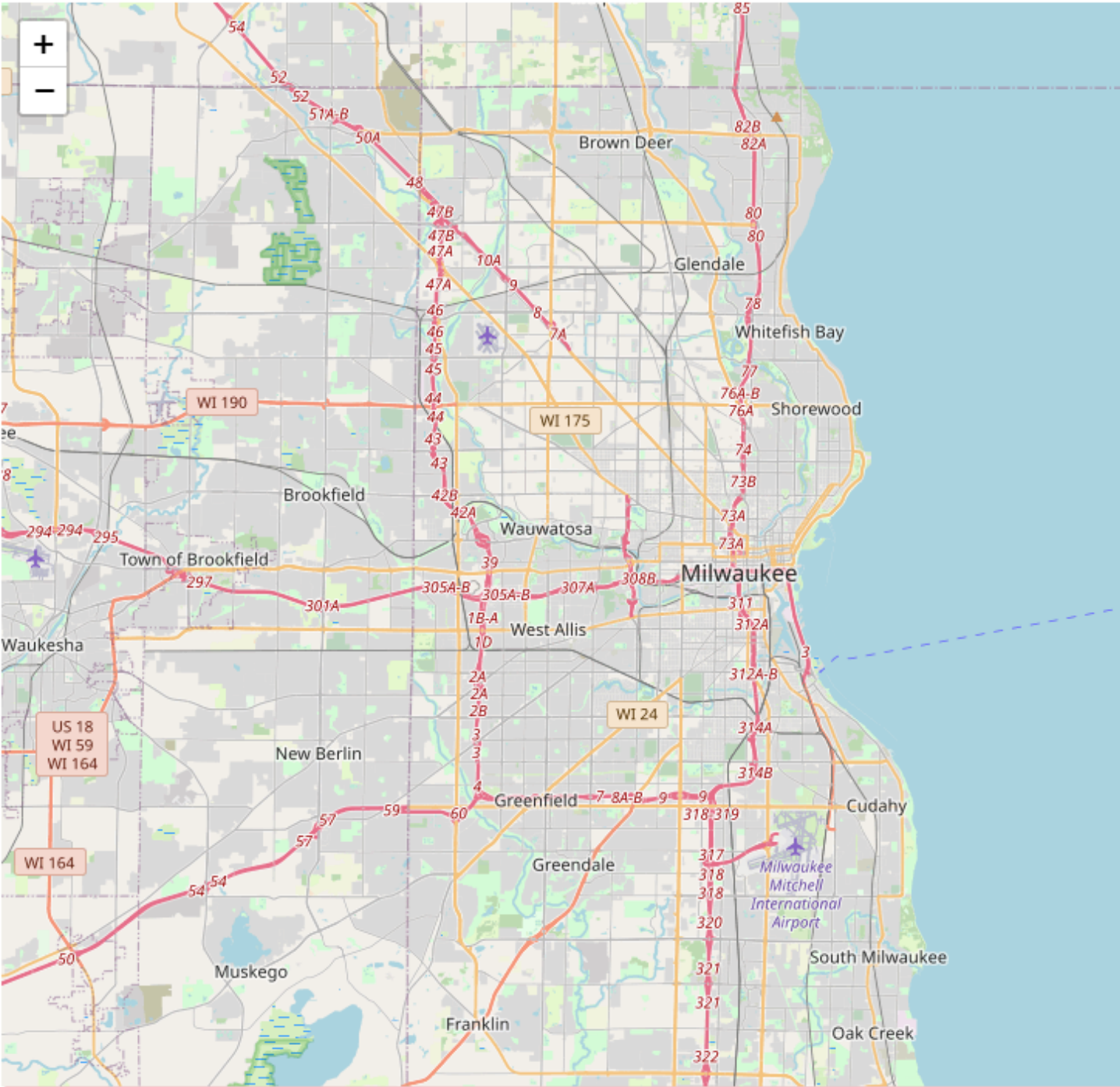


(<https://msoe.instructure.com/cou>

Route:

Start

Stop



Error response for route: asdf: No data found for parameter

Simulated Error for Route 1000:

<

<https://msoe.instructure.com/courses/9417/modules/items/373694>

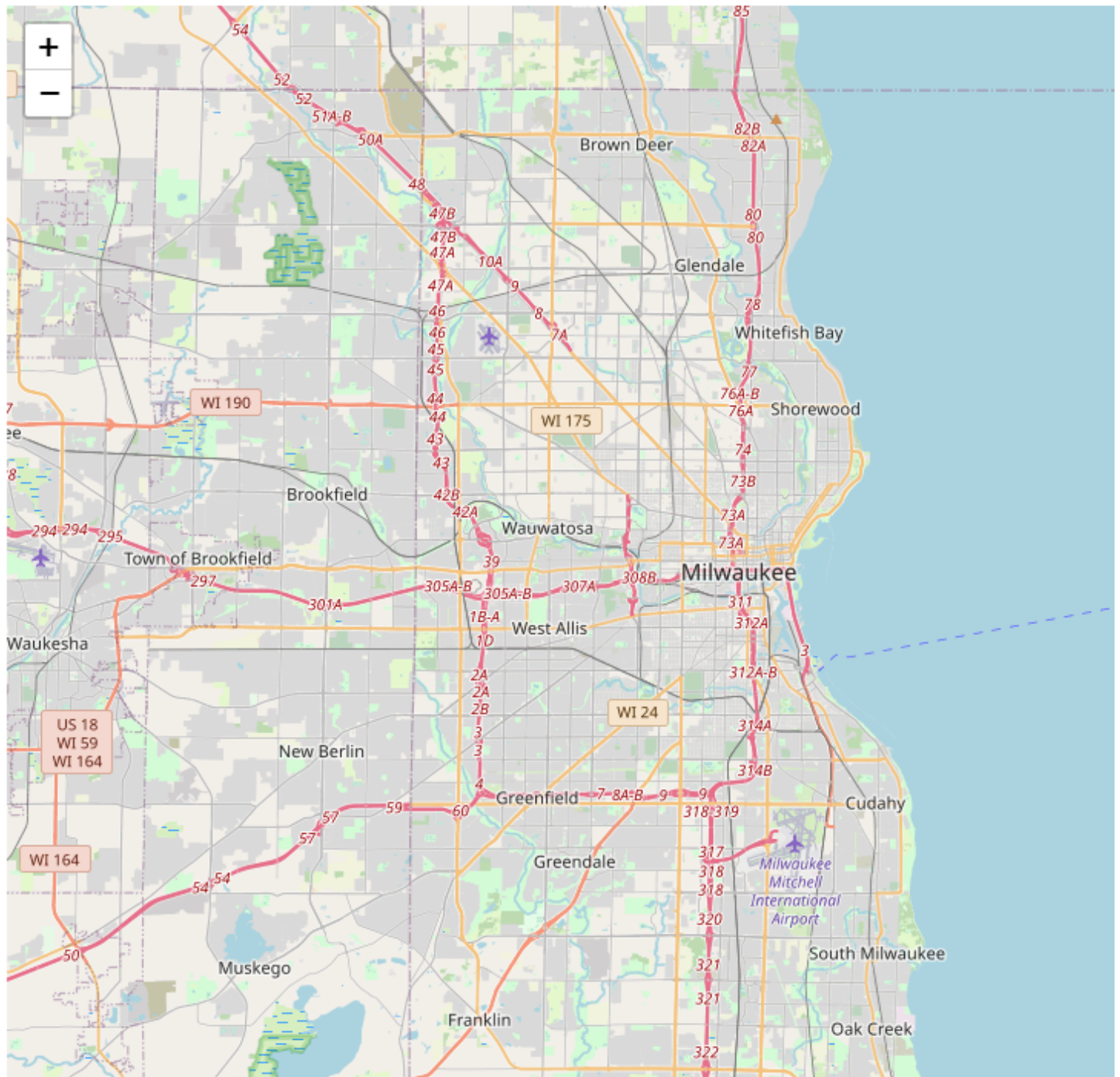
>

<https://msoe.instructure.com/cou>

Route: 1000

Start

Stop



Error: Server Error; IOException during request to ridemcts.com: Simulated server error during request

Extra Credit



(<https://msoe.instructure.com/courses/9417/modules/items/373694>)



Assignment

(https://msoe.instructure.com/courses/9417/assignments/110671?module_item_id=373791)

- Implement custom icons for the bus markers in place of pushpins (the default) representing the location of buses
- Delete the markers from the map when the "Stop" button is pressed
- Show only the last 10 markers (delete older ones) as the buses progress through their routes
- Add two more features of your own design (with prior approval from your instructor)

Deliverables

Test your program thoroughly! Be sure to test your application for various situations (bad key, bad route, AJAX call failure, etc). Make sure your implementation does not generate runtime errors, and presents appropriate error messages to the user.

You will need to include all HTML, CSS, and JS file(s) in your submission. Make sure your name, lab name, and section number are in comments on ALL file(s). Make sure to put your name in the TITLE tag of the HTML file.

ALSO, include a text file report containing:

- Introduction - a description of the lab in your own words
- Summary - a reflection of what you learned in this lab
- Suggestions - a list of suggestions for improvement and/or what you enjoyed about this lab

Make sure your JavaScript code is commented thoroughly. Create a zip file of all files and submit to Canvas before the due date.

NOTE: do NOT include any WebStorm specific files (e.g. your .idea directory). Only submit the files as specified.



✓ View Rubric

(<https://msoe.instructure.com/courses/9417/modules/items/373694>)

(<https://msoe.instructure.com/cou>)

Lab 5 Rubric		
Criteria	Ratings	Pts
Introduction view longer description		/ 2 pts
Summary view longer description		/ 2 pts
Suggestions view longer description		/ 6 pts
Code Structure view longer description		/ 10 pts
Functionality view longer description		/ 35 pts
Key Checkoff view longer description		/ 5 pts
Extra Credit view longer description		/ 10 pts
		Total Points: 0


Choose a submission type

<


>

(<https://msoe.instructure.com/courses/9417/modules/items/373694>)

(<https://msoe.instructure.com/cou>



Webcam



Canvas Files



Choose a file to upload