Make a new document using the naming convention HWFinal-MUid.docx (example: HWFinal-johnsok9.docx).

Objective: The objective of this exercise is to:
Use all of the skills learned in the entire course.

Submit: screenshots, your word document, html files and links to your web page(s)

You **may** discuss the concepts with your fellow students
You **may not** show or share code with your fellow students
You **may not** show or share code with internet sources
You **may** discuss this with your instructor or TA.

NOTE: This assignment includes a mandatory presentation by you to the Professor, along with submission to canvas. Any assignments which are not presented will receive Zero for the entire assignment. This must be working on your ec2-instance.

Part #1: Create GitLab project

Estimated time: 10 minutes

Exercise:

- Prep for gitlab
  - Ec2 as user ec2-user:
    - mkdir ~/.ssh
  - on ceclinux:
    - Copy your ~/.ssh/id_ed25519 into ec2 at the same location (use scp)
    - scp ~/.ssh/id_ed25519 ec2-user@ip:.ssh
  - Ec2:
    - chmod 700 ~/.ssh
    - chmod 600 ~/.ssh/id_ed25519
- Gitlab
  - Create your final project in /var/www/html
  - Clone it into your cse383_projects in git

Part #2: Client Front End

Estimated time: 2-3 hours

Exercise:

The first part will be a multi-page/multi-tab web client using html, JavaScript, bootstrap and CSS. You must make a good looking, pretty, responsive, web page(s) using bootstrap for formatting and your own CSS to make it look great. The page must degrade nicely which I will verify by shrinking it from a large desktop so that it fits down to phone size. Degrading means it must change, for example from 3 columns down to 1.

Note: Everything you write must use the following (from the exam)

- Concepts
  - Html
  - CSS
  - JavaScript
  - Bootstrap (must be responsive)
  - jQuery
  - Ajax
- Style
  - Good looking using CSS
  - **All CSS, ajax, JavaScript must be external files**
  - Every page (including the menu) must include pictures that integrate well into the web page
- Location
  - All code MUST be working on your ec2 instance http server, using your dynamic dns
  - All code must be in git
- Ajax
  - **You are to write all code using the jQuery Ajax routines as done in the lecture, labs, and homework. Some examples you see will use other methods, notably the fetch command. You may use those as an example, but you MUST use the $.ajax call for credit.**

The menu must provide the following items.

- Menu of pages you are providing (with a mechanism to switch pages) **You are to research and decide on your own mechanism** to provide the front-end menu of pages.
  - This was not covered in class. However, you have already seen web sites for research which included them.
- Pages
  - Landing page (default page)
    - Your information (name, class, assignment etc…)
  - Directions
  - Directions History and search

-

Part #3: Mapping Directions (using mapquest)

Estimated time: 3-10 hours

Exercise:

- Use the mapquest api you created previously
- Make a good-looking, responsive, page which lets the user enter a from and to address (see: https://developer.mapquest.com/documentation/common/forming-locations/ for details on what can be entered)
- Call the mapquest directions api (https://developer.mapquest.com/documentation/directions-api/route/get/)
    - Display all returned maneuvers with narratives, distance, time and thumbnail of the map image (mapURL)
- Call the elevation chart api
    - Display one elevation chart using the two addresses as 400x300 elevation chart
        - https://developer.mapquest.com/documentation/open/elevation-api/elevation-chart/get/
- **Store all information about the request (and response(s)) using the php rest server from lab13**
    - **Store your own JavaScript object by creating the json version of the object and storing that in the value field**

Part #4: Request History

Estimated time: 2-6 hours

Exercise:

Provide a page where the user can enter a date and max number of lines and then receive a list of all directions requests made on that date in a nicely formatted output (table), showing all available data, with not more than the requested number of lines.

- You will be calling the updated rest interface on your ec2 instance using ajax
- *https://uniqueID.aws.csi.miamioh.edu/final.php*
    - *Method:getLookup*
    - *Date:yyyy-mm-dd*
- You must show the correct number of returned results (based on the input page)
- For each returned request, use the json object you sent in part 3 to show the following:
    - Date
    - Time
    - From
    - To
    - Number of maneuvers
- The user should be able to click on one of the lines (make something selectable or add a button)
    - This should then display the same information as in part 4, but using the data from your json object
        - Do **not** call mapquest again. You must have all of the data in the json object you stored in part 3. If not, update part 3 to store new data.

Part #5: Write Word Document

Estimated time: 45 minutes

Exercise:

- Always make sure it includes your name, class, assignment etc…
- Explain (relating to this assignment and the entire course):
- This must be in full paragraphs with full sentences for full credit.
    - what worked
    - what didn't
    - how long it took
    - What you liked
    - What you didn't

SUBMIT:

- Word Document
- **Mandatory Presentation of working system in zoom session**
- Upload all code
- Git all code
- Paste the URL of your menu page into the submission as a comment