Requirements

This week you will write single page web application that will receive incoming POST and GET requests. For the "single page", it means you should use one URL for both types of requests. If it is a POST request it should say at the top of the page, in an H1 tag "POST Request Received". If a GET request is received it should say "GET Request Received" in the same H1 tag.

Below that you can, at your discretion, create either a list or a table which clearly shows all parameter names and values which were sent in the URL query string for both a GET and POST request (you can still send stuff in the URL when making a POST request).

Below that, if it is a POST request, you should list, again either in a list or a table, all the property names and values that were received in the request body. It should be able to accepter either a well formatted URL encoded query string (x-www-form-urlencoded) or JSON data.

Deliverables

You should submit a single zip file containing all of your source code.

As an exception to the general call rule you can copy and paste the code from the lecture without citing or explaining it. **This does not apply to the routes used for the GET and POST submissions.** So all the boilerplate code about setting up handlebars, handling 404's, launching the app can all be copied because it is really boring and you will be using it a lot. The handlers for the routes providing the data should not be copied from the lectures.

This should include the .js files, your package.json file, and the directories containing your views, templates and static files. Do not include the node_modules directories in this zip file.

In addition include a **URL as a comment on Canvas added to your submission** which links to a live, functioning version of your page. It is your responsibility to keep this running until you get a posted grade back. When you are done and happy with your code, I recommend you start a new instance of node.js using the forever application on a different port than you use for development. Then never touch it again until we are done grading. If you have questions of making it running on the engineering flip, please check out the **Week one** content and assignment. (If on engineering flip, the URL should be something like **http://flipX.engr.oregonstate.edu:YYYY, do not use localhost**)

To test (**only POST**) key-value pairs **sent in the body,** you need to use an HTTP POST request. You could do this via a Javascript call, but it might be easier to use a Chrome plugin like Advanced Rest Client (Links to an external site.)Links to an external site. (**ARC**) to get a GUI interface you can play with.

If it is not up and running when we try to grade it then there will be a flat 5% deduction and we will contact you to get it up and running so we can grade it.

Note this is graded like a full HW **assignment** so it is weighted more heavily than an activity. That said, if all goes well it should not take that long. Do not expect the coming weeks assignments to be as quick. I want to make sure to leave some buffer here for problems that will arise given you are brand new to server side programming.