Documentation Requirements

GPA Calculator

## What your site does and its purpose.

As someone who is very concerned about his GPA and wants to keep track of where I am, I created an Excel spreadsheet which keeps track of and calculates my GPA.

I’ve thought of building a web application that works with SQL, and when we started working in the course I thought it would be nice to make a site that is restricted, which one can log into, enter your courses and have a live representation of your overall GPA.

This site allows a user to register and then log in. The user can then select the courses by course ID and add them to their list. You can switch between possible states and the GPA is calculated based on the courses that are marked as Completed. The GPA is calculated on the 4.0 scale from a letter grade.

Courses are added by default without a grade with a “In Progress” status. You can then select the courses, change the status and grade and update, which will update the data base, calculate the GPA and display the new value.

All the data in the program is stored on a SQL database and is permanent. New users are created and require a password to log in, however you can still navigate to data pages with the direct link and parameters (because this is a requirement) and as such I would not suggest it be used with official information until it is secured.

## What part of your app does the API handle?

The entire site (two html pages) are handled by the API. It includes authentication, dynamic form building and updating, and SQL data queries. There are multiple JavaScript files that handle different aspects of requests:

* authServer.js handles user login and user creation
* cssResponses.js handles all CSS file requests (there are 4 different CSS files)
* dbOperations.js handles all database related tasks including:
  + INSERT
  + SELECT, and
  + UPDATE
* gpaPageManager.js handles the GPA html page generation and operations
* htmlResponses.js handles all html file requests
* jsonResponses.js handles all json data transmissions
* jsResponses.js supplies the gpaPageManager.js file to the gpa html page (it is the bundle file for the second html page)
* server.js runs the server and interprets requests

## What went right and what went wrong?

As someone who has never worked with SQL on a request/response basis it took me a long time to figure out how to manage the queries and requests with callbacks. Not to mention that I’ve never used callbacks before and it took me a long time to actually understand how they work also.

Oh… and I think ESLint has some ridiculous rules. Things like not allowing methods to call methods that are below them (the whole “calling out of sequence” or “calling before assignment” thing). And not allowing normal loops without jumping through hoops. I had more than one function where I had to build an entire true/false tree of values because it didn’t allow me to use a for-loop, and you can’t break out of a Object.values().forEach() loop.

## If you were to continue, what would you do to improve your app?

At the moment you cannot add more courses to the list unless you have access to, and know how to use SQL. All the IGME courses are listed (I added them manually during testing) but I don’t have a form to add new courses from the client side.

This causes a limitation because I don’t have the time to add all the courses RIT has by hand, and people can’t add their own courses at this time.

I would like to add a third html page that allows users to add their own courses. The SQL and request/response requirements won’t be that hard (I’ve streamlined the process) however the conditions that would be required to ensure people add all courses using a predefined style, and don’t duplicate courses would require some additional work.

## How did you go above and beyond?

* I have two html pages that connect to the same node.js server. Both pages use requests and responses and both pages use SQL as a backend data store.
* There is user creation and authentication included in the project.
* I’m also using the Heroku ClearDB MySQL database for storage, so all data is persistent.
* I created a fully relational database to house the data and have extensive queries to ease the process.

