User Stories to Work on for the NEXT GROUP

* As an admin, I can add curricula with skills
* As an admin, I can add a building/location picture
* As a trainer, I can add a profile picture
* As a trainer, I can add time off to the shared Google Calendar and a corresponding unavailability will be saved to my set in the database.
* As a user, I can create at least one custom naming convention (template is already on the settings tab)

Known Issues / Bugs

* Cannot currently edit buildings or locations which own batches down the chain.
* We should have a text size by default for all our regular tooltips off of buttons instead of having to add md-title to each of them.
* Signing in is not implemented (should be passed a token by Parasol)
* Differentiation therefore between admin and trainer views is not implemented. We were thinking to just ng-hide or show based on user credentials, as a one-page application
* Settings are global in the db. They should be unique to each user.
* Each feature has not been put through every single test case, as we had no automated testing going on throughout production.
* If you go to the batches tab, then settings, change the default location and go back, you have to refresh the batches page for it to take effect.
* In the Settings page, if you select a location, its inactive buildings will be possible selections for default.

\*\*Ideally, these should be listed as issues on GitHub\*\* ← yessss

Notes for NEXT GROUP

There are several layers of this application. Let’s start at the front end.

The practical entry point is index.html, in the resources folder. This is where all the .js references, stylesheets, and the menu bar is loaded.

Each tab in the application may display one or more dialog or toolbars, all of which are also found in the resources folder.

The js controllers, simply in the js folder in resources, are practically the main business logic layer. They store all necessary data and operations you can perform when viewing content on certain pages.

In the same js folder are the services. These js services are called from the js controllers to retrieve or store information via the *java* controllers (each named with a Ctrl suffix), by the use of DTO’s. Data Transfer Objects, in this case, can be, for example, created using javascript objects in the controller, passed into the service in the matching format of the corresponding DTO, passed into the Java controller, and finally into the Java DaoService to do business with the database. So it’s like this:

Java Side:

POJO, DTO, DaoService, RESTCtrl

JS Side:

Service, Controller

And the flow is:

View HTML -> Use a JS Controller -> JS Service -> Java RESTCtrl (passing in js object as predefined DTO) -> Java DaoService

Each table object has its own handlers beginning at JS Service down to DaoService.

*Shared link* in Google Docs, for real-time editing viewable / editable by all: <https://docs.google.com/document/d/1GSVoI3Yecx0CJN0U9mzHmL3YpFHPfEMKLnUnsB7okPo/edit?usp=sharing>

Make sure to update the documentation within the project folder as well after significant changes have been made, preferably by one person.