Report: HTTP Requests / Routes
In the report, include a list of HTTP routes implemented in the server, and the mock server method that they replace. For each, briefly describe:
• The verb, target, and body of the request.
– Example: POST /feeditem [feeddata], where [feeddata] is a JSON object describing a Feed Item.
• What the HTTP request does, and what mock server method it replaces.
• What resources the HTTP request creates/modifies/etc.
– Examples: “Updates /user/:userid”, “Creates a new /user”. . .
• Who is authorized to use the request
– Example for PUT /user/:userid userdata: “A user with the specified :userid can issue this request. Administrators can also make this change on any :userid.”
– Example for POST /feeditem feeddata: “The body of the HTTP request contains an”author" field containing a user ID. The requester must have the same user ID."
– If you have a hard time explaining it generally, feel free to include examples. Example: “A user with ID 4 can issue a PUT /user/4, but a user with a different ID cannot.”

Report: Special Server Setup Procedure
If your server has any advanced features that require additional setup besides npm install and node src/server.js, include a section in the report that describes how to set up these features. Include details on how to tell if the feature is working properly.

Report: Individual Contributions
Include a section that describes each startup founder’s contribution. Each startup founder should be responsible for at least one product feature and its HTTP route(s). Name the feature and the HTTP routes that the founder implemented / was responsible for.

Report: Lingering Bugs / Issues / Dropped Features If your application has any lingering bugs, issues, or missing/dropped features, include a section listing these.

HTTP Requests / Routes:

GET /user/:userid - This request gets a JSON object with all of the user’s important information, such as majors, courses they’ve taken, their account info, their saved graphs, and anything else we need to display to help them use the app. In the mock server, we had this as getUserData2, which was simply the asynchronous version of getUserData. Only users with the specified :userid can issue this request. Administrators can also see this information through the request.

PUT /user/:userid/majortoshow/:majorid - This request adds a new “shown major” to the user item in the database. Only a “shown” major is displayed on the graph on the homepage in the form of all its courses. The method modifies /user/:userid and adds a major with :majorid to the

“shown_majors” variable. It replaces the mock server method “addShownMajor”. Only the user with :userid and administrators can make this change.

PUT /user/:userid/minortoshow/:minorid - This request is exactly the same as PUT /user/:userid/majortoshow/:majorid, except it adds a shown minor to the user’s data.

PUT /user/:userid/courses/:course - This request adds a course to the users course history. Only the user is authorized to change this information. This is a feature that wasn’t included in the last submission so it doesn’t replace anything.

PUT /user/:userid/courses/:course/nextsem - This request adds a course to the users courses for next semester. Only the user is authorized to change this information. This is a feature that wasn’t included in the last submission so it doesn’t replace anything.

DELETE /user/:userid/majortoshow/:majorid - This request takes out the major with :majorid from the “shown_majors” variable in the user with :userid. It is the opposite of the PUT request. This then takes the major out of the main graph on the homepage to allow the user to reduce possible visual clutter. It replaces the mock server method “subtractShownMajor”. It updates /user/:userid by taking a shown major out. Only administrators and the user with :userid can issue this request.

DELETE /user/:userid/minortoshow/:minorid - This request is the same as DELETE /user/:userid/majortoshow/:majorid, except it takes out a shown minor, rather than a major, from the user with :userid.

GET /user/:userid/page - This request gets a JSON object with save pages information. Only user with the specified :userid can issue this request. Administrator can also see this information through the request

DELETE /user/:userid/page/:pageid - This request deletes JSON object that matches with page id. Only user with the specified :userid can issue this request.

DELETE /user/:userid/courses/:course - This function deletes a course from a user’s course history. Only the user is allowed to use this function. This is a new feature so it doesn’t replace any previous functions

DELETE /user/:userid/courses/:course/nextsem - This function deletes a course from a user’s courses for next semester. Only the user is allowed to use this function. Again, this is a new feature so it doesn’t replace any previous functions

POST /savedgraph - This function adds an image representation of the main app’s graph to a user’s list of saved graphs.

POST /feedback - This function adds the ability for any user to send us feedback on the application.
GET /feedback/:userid - This function allows us to read any feedback posted by users. Currently, if the user is an authorized admin, they may read feedback, however, the only way to do so is through postman.

Special setup:
Clicking the “Save Progress” button on the homepage prompts a dialog box - if this does not pop up then you might need to enable pop ups in your browser.
Cytoscape.js is third-party code used to help render the homescreen, although everything necessary to run it should already be set up.
You can get the feedback (which is sent by users through the about page) as an admin in Postman with GET localhost:3000/feedback/1 and Authorization = “Bearer eyJpZCI6MX0=” (Since user 1 is an admin).
The reset database button is on the right side of the top navbar.

Individual Contributions:

Mike Keegan - Implemented accurately displaying the information on the main page, which needs to get the user’s data to display and allow the user to add/subtract majors or minors from the display. Added routes to get user data, put shown major, put shown minor, delete shown major, and delete shown minor. Added the Error Banner and cleaned up errors from last submission, including restructuring the main sidebar to properly get info from server asynchronously.

Anthony Depace - implemented the functionality of the download PNG button on main page to export the graph to a png image. Added the route for saving a graph to the server (by clicking the ‘save progress’ button on the main app).

Geon Yoon - implemented the functionality of everything in savePage page such as search bar, displaying save pages, deleting certain pages, and minimize & enlarge the image. Added routes to get page data, and delete the specific page.

Andrew Knowles - implemented course details so that they load data from the server using routes already created. Made routes to add and remove courses from a user’s course history. Got course history to also load data from server. Cleaned up code and deleted unused functions and components.

Ben Reul - fixed css issues on Profile.js and Save Page. Changed Profile page layout and removed unnecessary elements.Rewrote navbar code to use database to show user name. Made routes to add and view feedback posted on the about page.

Lingering bugs/issues/dropped features:

The interactive graph on the main page gets some bugs if the same class is in multiple majors. This might be able to be fixed, but not without changing the way the data is loaded. We will look into it, but for now we simply keep classes for majors limited to their respective department so that there is no overlap. The text version of the main graph has been omitted, as the features that were once lacking in the graph representation are now implemented and the text is no longer needed. In order to show a course as “taken”, you must also remove it from “take next semester”. This is to benefit those who might need to retake a class, but the way we implemented it might be confusing for some. Course information page has issues displaying correct information without refreshing page. “Similar classes” in the course details page has not been implemented yet. The drop down menu to filter by year in course history was taken out. The change password text boxes and button have not yet been implemented since we are still waiting to learn about authentication and security.