This document serves as a dissection of the current routes found in the Database Breakdown branch. Each route section shows the purpose of the route, what is needed for the route to work, error codes, and results.

/health - Health Route

Return whether port is active

No params needed.

Responds with data array: {status, port} and a 200 on success.
Returns a 500 for an internal server error.

Session Route:

GET

/session/ - Log In

Given an email and password, return an access token

Params:
Body:
email, this refers to user email
password, this refers to user password

Responds with an access token and a 200 on success.
Return a 400 if email/password is missing with msg: “[Email/Password] is required”.
Returns a 404 if user is not found with msg: “User not found”.
Returns a 401 if password is incorrect with msg: “Invalid password”.
Returns a 500 for an internal server error with msg

POST

/session/ - Create a User

Given a name,email,and password, return newly created user info

Params;
Body:
name, this refers to user name
email, this refers to user email
password, this refers to user password

Responds with data array: {name,email,password} and a 201 on success.
Returns a 400 if name/email/password is missing with msg: “[Name/Email/Password] is required”.
Returns a 409 if user already exists with given email with msg: “User already exists”.
Returns a 500 for an internal server error with msg.

User Route:

GET

/user/self - Get User Info

Given an access token, return user info
Route requires JWT Verification

Params:
Headers:
authorization: this refers to the header where the access token is typed in

Responds with data array: {name,email} and a 200 on success.
Returns a 401 if there is no authorization header with msg: “Empty authorization header and token”.
Returns a 401 if user did not insert access token with msg: “Enter a token”.
Returns a 403 if user entered an invalid token with msg: “Invalid token”.
Returns a 403 if user does not have the proper role with msg: “User does not have required claims”.
Returns a 400 if email is missing with msg: “Email is required”.
Returns a 500 for internal server error with error msg.

/user/allCourses - Get All Available Courses

Route requires JWT Verification
Given an access token, return all available courses for user

Params:
Headers:
authorization, this refers to the header where the access token is typed in

Responds with data array: Course{name} and a 200 on success.
Returns a 401 if there is no authorization header with msg: “Empty authorization header and token”.
Returns a 401 if user did not insert access token with msg: “Enter a token”.
Returns a 403 if user entered an invalid token with msg: “Invalid token”.

Returns a 403 if user does not have the proper role with msg: “User does not have required claims”.
Returns a 404 if no courses are available with msg: “No courses found”.
Returns a 500 for an internal server error with error msg.

/user/transcript - Get User Transcript

Given an access token, return all courses user is enrolled in
Route requires JWT Verification

Params:
Header:
authorization, this refers to the header where the access token is typed in

Responds with data array:
Enrollment{enrollment_id,student_email,professor_email,course_name} and a 200 on success.
Returns a 401 if there is no authorization header with msg: “Empty authorization header and token”.
Returns a 401 if user did not insert access token with msg: “Enter a token”.
Returns a 403 if user entered an invalid token with msg: “Invalid token”.
Returns a 403 if user does not have the proper role with msg: “User does not have required claims”.
Returns a 400 if email is missing with msg: “User email is required”.
Returns a 404 if user is not enrolled in any course with msg: “No enrolled courses”.
Returns a 500 for an internal server error with msg.

POST

/user/newCourse - Enroll in a Course

Given an access token and course name, return created enrollment
Route requires JWT Verification

Params:
Headers:
authorization: this refers to the header where the access token is typed in
Body:
courseName, this refers to name of course

Responds with data array: {enrollment_id,student_email,professor_email,course_name} and a 201 on success.
Returns a 401 if there is no authorization header with msg: “Empty authorization header and token”.
Returns a 401 if user did not insert access token with msg: “Enter a token”.
Returns a 403 if user entered an invalid token with msg: “Invalid token”.
Returns a 403 if user does not have the proper role with msg: “User does not have required claims”.
Returns a 400 if email or course name is missing with msg: “[User email/Course name] is required”.
Returns a 404 if courseName does match any existing course with msg: “Course does not exist”.
Returns a 409 if user is already enrolled in course with msg: “Enrollment already exists”.
Returns a 500 for an internal server error with msg.

DELETE

/user/dropCourse - Drop Course Enrollment\

Given an access token and course name, delete user enrollment to course
Route requires JWT Verification

Params:
Headers:
authorization: this refers to the header where the access token is typed in
Body:
courseName, this refers to name of course

Responds with a 204 on success.
Returns a 401 if there is no authorization header with msg: “Empty authorization header and token”.
Returns a 401 if user did not insert access token with msg: “Enter a token”.
Returns a 403 if user entered an invalid token with msg: “Invalid token”.
Returns a 403 if user does not have the proper role with msg: “User does not have required claims”.
Return a 400 if userEmail/courseName is missing with msg: “[User email/Course name] is required”.
Return a 404 if courseName does not match any existing course with msg: “Course does not exist”.
Returns a 404 if user is not enrolled to this course with msg: “User is not enrolled to this course”.
Returns a 500 for an internal server error with msg.