BACKEND ENDPOINT GUIDE

Everything returns a json object, aside from the particular data you want it also returns with a integer field “status”
0: success, go check the rest of the fields for your data
-1: failure, you didn’t put in valid data or put in improperly formatted data
-2: failure, you didn’t provide all the needed fields

User/Class data field types:

- email: string
- id: integer
- role: boolean

- /backdoor/
    - Not intended for use outside of testing, allows use of most POST and DELETE functions without authentication
    - Takes fields “table” and “data”
        - “Table” holds “users”, “classes”, “userclasses”, or “class-students”
        - “Data” holds an array of data used for the functions. Standardized so I can copy-paste easier
            - [user, class, role, schedule]
    - GET
        - “Users”
            - Runs getUserDetails() and returns with the same format as usual
        - “Classes”
            - Runs getClassDetails() and returns with the same format as usual
        - “Userclasses”
            - Returns {"data":getUserClasses(req["data"][0])}
        - “Class-students”
            - Returns {"data":getClassMembers(req["data"][1])}
    - POST
        - “Users”
            - Adds a user with the email and role specified in “data”. Doesn’t use username validation and password is “1!Password”
        - “Classes”
            - Adds a class with the id specified in “data” with the name “TESTCLASS”
            - Doesn’t add any members
        - “Userclasses”
            - Adds the specified user to the specified class with the role specified in “data”
    - PATCH
        - Unimplemented, returns {} with code 400
    - DELETE

        - “Users”
            - Returns {"status":deleteUser((req["data"])[0])}
        - “Classes”
            - Returns {"status":deleteClass(req["data"][1])}
        - “Userclasses”
            - Returns {"status":leaveClass(req["data"][0], req["data"][1])}

- /register
    - POST
        - Takes fields “email”, “password”, and “role”
        - Returns with “error” field if certain conditions are met
            - Email doesn’t end with “@umass.edu”
            - Password
                - Less than 8 characters long
                - Doesn’t contain a uppercase letter
                - Doesn’t contain a lowercase letter
                - Doesn’t contain a number
                - Doesn’t contain a special character
                    - [!@#$%^&*(),.?\":{}|<>]
- /login
    - POST
        - Takes fields “email” and “password”
        - Returns with {"status": "logged in", “role”:**USER_ROLE**} on success
        - Returns with {"status": "invalid credentials"} and HTML code 401 on failure
- /logout
    - No arguments
    - Ends the current user session if possible, returns code 401 when not logged in
- /users/
  Check for “email” field in json input
    - GET
      Check for “data” field in json input
        - Check if “data” field is:
            - “profile”
                - Returns {“email”:string, “name”:string, “role”:boolean, “schedule”: int[5][12]}
            - “schedule”
                - Returns {“schedule”:int[5][12]}
            - “name”
                - Returns {“name”:string}
    - PATCH
        - Check for “schedule” field in json input
            - If true, attempts to update the user’s schedule, takes int [5][12] array as input

            - Adds field “schedule” to the return object, 0 if successful, -1 if not
            - Sets “status” to -1
    - Check for “name” field in json input
        - If true, attempts to update the user’s name
            - Adds field “name” to the return object, 0 if successful, -1 if not
            - Sets “status” to -1
    - Check for “password” field in json input
        - Check for “newPassword” field in json input
            - If false
                - Set “password” = -2
                - Set “status” = -2
            - If true
                - Run password validator on “newPassword”
                    - If invalid
                        - Set “password” = -1
                        - Set “status” = -1
                        - Set “message” to the error message the password validator returns
                    - If valid
                        - Attempt to change the password to the new one
                            - If “email” and/or “password” are incorrect
                              Set “password” = -1
                              Set “status” = -1
                              Set “message” = "Incorrect User or Password"
                            - If success
                              Set “password” = 0
- DELETE
    - Nothing special, “status” is 0 on success, -1 on failure

- /classes/
  Check for “id” field in json input
    - GET
      Check for “data” field in json input
        - Check if “data” field is:
            - “all”
                - Returns {“classid”:int, “name”:string, “schedule”:int[5][12], “officehours”:int[5][12]}
            - “name”
                - Returns {“name”:string}
            - “schedule”

                - Returns {“schedule”:int[5][12]}
            - “officehours”
                - Returns {“officehours”:int[5][12]}
    - POST
      Check if user session is a teacher
        - Returns {“status”:-3, “message”:”begone student”} if not
      Check for “name” and “email” fields in json input
        - “status” is 0 on success, -1 on failure
        - Adds the current user to the class as a teacher automatically
    - PATCH
        - Checks if user is teacher in the specified class
            - Returns {“status”:-3} if they aren’t
        - Check for “schedule” field in json input
            - Attempts to update the field, takes int [5][12] array as input, 0 on success, -1 on failure
        - Check for “name” field in json input
            - Attempts to update the field, 0 on success, -1 on failure
        - Check for “officehours” fields in json input
            - “officehours” is used as the number of office hours per day
            - Recalculates office hours
            - Adds field “officehours” to return, contains updated set of office hours(int[5][12]) on success, -1 on failure
    - DELETE
        - Checks if user is teacher in the specified class
            - Returns {“status”:-3} if they aren’t
        - Check for “password” field in json input
            - If not, return {“status”:-2}
        - If password doesn’t match the current password
            - Return {“status”:-1}
        - Returns {“status”:0} on success
- /users/classes/
  Checks if user is logged in
      Returns {“status”:-3} if not
    - GET
        - Returns {“classes”:[][int, boolean]} of the current user
    - POST
      Check for “id” field in json input
        - Regular status return, 0 on success, -1 on failure
        - Adds the current user to the specified class as a student
            - Users must be promoted to teacher by an existing teacher to prevent/reduce potential sabotage by other staff
    - PATCH
      Check for “email”, “id”, and “role” fields in json input
        - Returns {“status”:-1} if not present

        Checks if current user is teacher in class

        - Returns {“status”:-3} if not
        - Attempts to change the specified user’s role in the class
            - Value of status depends on if that user exists in that class or not

    - DELETE

      Check for “id” field in json input

      Check for “email” field in json input

        - If “email” exists, check if user is a teacher in the class
            - If false, return {“status”:-3}
            - If true, return 0 on success, -1 if the target user isn’t in the class
        - If “email” doesn’t exist
            - return 0 on success, -1 if the current user isn’t in the class

- /classes/students/

  Check for “id” field in json input

  Checks if the current user is a teacher in the specified class

    - GET
        - Returns {“members”:[][string, boolean]} of the specified class