FE: Evans-classroom.com
HTML form: {email: matthew@gmail.com, password: 123…}
User hits submit->

HTTPS POST request (req) to myclassroom.com/users
    Body: { “email”: “matthew@gmail.com”.... }
    Headers
        Authorization: JWT

BE: Myclassroom.com
Gets the request to myclassroom.com/users
Reads all the info
Authentication (if required for this route)
Does stuff (creates a user)

HTTPS response (res)
    Body: { “errors”: “” } or { “email”: “”, “firstName”: “Matthew” }
    statusCode: 200 - 299 good, 201 - created
        400 - bad request, 401 - unauthenticated, 403 - unauthorized, 404 - not found
500 internal

HTTPS requests
    GET - get info
    POST - create / PATCH
    PUT - update
    DELETE - delete

www.myclassroom.com

/users

POST: /users

Body: { “email”: “”, “password”: “”, “confirmedPassword”: “”, “firstName”: “”, “lastName”: “” }
    Note: we can add more information to the user account if we please
User Action: create account submission
Backend Action: attempt to create an account, if successful send email confirmation
Backend Response:

- If successful, create & send user token with 200
- Else if the email is already in use, send a 400 with that info
- Else if some other validation fails, send a 400 with validation error

Routing:

- If 200, store the token in cookies and send to /courses (should be empty)
- Else, display the error and keep all values on the form

Other: frontend should perform all the validations on matching password and confirmed password, valid email format, etc. The only thing it can’t check is that email is in use

/password

PUT: /users/password?email=””

User Action: request password reset
Backend Action: if the email is attached to an account, send an email to it with OTP reset code
Backend response:

- If account not found for that email or email sent, return a 200
- Otherwise, send a 500

Routing:

- If 200, send to the password OTP entry page
- Else, display an error message that the server had an issue and to try again

/authenticate

POST: /users/authenticate

Body: { “email”: “”, “password”: “” } - optional
User Action: click on login button or reload page
Backend Action: authenticate the user with credentials provided or use the token to authenticate
Backend Response:

- if authenticated, create a user token and send back with 200
- Else, send back a 404

Routing:

- If 200, store the token in cookies and send to /courses
- Else, display login error and stay on the page. Clear password, keep email

/:user_id

GET: /users/:user_id

User Action: account page
Backend Action: authenticate token and get that email, firstName, lastName
Backend Response: send the information if authenticated, otherwise send a 404

PUT: /users/:user_id

Body: { “email”: “”, “oldPassword”: “”, “newPassword”: “”, “confirmedPassword”: “”, “firstName”: “”, “lastName”: “”, “emailConfirmation”: “” }
User Action: edit account submission

Backend Action: authenticate with the oldPassword, if provided. Or authenticate with the password reset field in the token. Authenticate emailConfirmation if present. Update necessary fields, assuming success
Backend Response:

- If authenticated and other information validation passes, return 200 with the new data (except password obviously)
- Else send a 400 bad request with any errors that were generated by the validation

Routing:

- if successful, return to user page with updated information
- Else stay on the page and display errors

Other: front end should validate a valid email address, new and confirmed password match

DELETE: /users/:user_id

User Action: request delete account from account page
Backend action: authenticate token & delete user (and associated data) if authenticated
Backend response: 200 if successful, otherwise 404 if unauthenticated or 500 otherwise
Routing:

- If 404 or 200, send the user to the home page
- Else stay on the page and say try again

/courses

GET /courses

User Action: home page
Backend action: authenticate token & get the user’s course (teacher and student logic different here - if student in a course, getting their sections returned, whereas teacher gets courses. If a user is a teacher in one course and a student in another, just send both back)
Backend response: 200 and their courses if successful, otherwise 401 (auth fail)
Routing: if 200, stay. Otherwise home page, no authentication

POST /courses

Body: { “name”: “”, “description”: “” }
User Action: creating a course
Backend Action: authenticate token & create the course & create enrollment for logged in user as teacher in the course
Backend response: If authenticated, return 201. Otherwise 400 (invalid fields)
Routing:

- If 200, send the teacher to that course page
- Else, send to the home page

POST /courses/join

Body: { “joinCode”: “” }
User Action: student form submission to join course with course code (actually a section code)

Backend Action
Backend Action: authenticate token & create an enrollment for the user in the section that has the code
Backend Response:
- If authenticated and the section was found, return 200
- If authenticated but the section was not found, return 404
- Else return 401

Routing:
- If 200, render courses page
- If 404, render not found error and stay on page
- Else send to the home page unauthorized

/:course_id

GET /courses/:course_id

User Action: click on a course
Backend action: authenticate token & check user is enrolled in the course (teacher and student logic different here. Teacher role gets course information & all the sections, whereas student just gets course information)
Backend response:
- If authenticated, return 200 & the data
- Else, return 403

Routing:
- If 200, render the page for that course
- Else, send to home page unauthenticated

PUT /courses/:course_id

Body: { “name”: “”, “description”: “”, “published”: true/false }
User Action: teacher is updating their course
Backend action: authenticate token & update the information
Backend response:
- If authenticated, return 200 & the data
- Else, return 403

Routing:
- If 200, render the page for that course
- Else, send to home page unauthenticated

DELETE /courses/:course_id

User Action: teacher is deleting a course
Backend Action: authenticate token & delete the course & all its data
Backend response:
- If authenticated, delete the course, all associations to it, and return 200
- Else, return 403

Routing:
- If 200, redirect to the /courses page
- Else, send to home page unauthenticated

/sections (teacher only)

POST /courses/:course_id/sections

Body: { “number”: “” } (optional)
User Action: teacher is adding a section to this course
Backend Action: authenticate token & create the section & associate any existing course lectures to the section
Backend Response:
- If authenticated & all creating goes well, return section information and 200 status
- If authenticated but number is taken, return 404 with error for bad section number
- Else return 403

Routing:
- If 200, show the courses page for the teacher
- If 404, indicate the error
- Else send to home page unauthenticated

GET /courses/:course_id/sections

User Action: teacher is getting all sections for a course
Backend Action: authenticate token & get all sections for course
Backend Response:
- If no sections under this course, 204
- If user is not a teacher, 403
- If database validation error, 400
- If everything is correct, 200 with all section information

Routing:
- If 200, show the sections to the teacher
- If 204, indicate that there’s no sections available to view
- Else send to home page unauthenticated

/:section_id

PUT /courses/:course_id/sections/:section_id

Body: { “number”: “” } (optional)
User Action: teacher is updating the section number
Backend Action: authenticate token & update the section
Backend Response:
- If authenticated & all updating goes well, return section information and 200 status
- If authenticated but number is taken, return 404 with error for bad section number

- If student return 403
- Else return 401

Routing:

- If 200, show the courses page for the teacher
- If 404, indicate the error
- Else send to home page unauthenticated

GET /courses/:course_id/sections/:section_id

User Action: teacher wants to enter a section in one of their courses
Backend Action: authenticate token & find the section & it’s lectures
Backend Response:

- If authenticated & session found, return the section, namely the number and joinCode with 200
- If not found, return 404
- If student, return 403
- Else return 401

Routing:

- If 200, render the section page
- Else if 404, stay on courses page
- Else send to home page unauthenticated

/lectures/:lecture_id

GET /courses/:course_id/sections/:section_id/lectures/:lecture_id

User Action: teacher wants to view the lecture, published or not - will determine what shows up
Backend Action: authenticate token, find the lecture, find the questions
Backend Response:

- If authenticated as teacher, returns the lectures with 200
- If student, return 403
- Else return 401

Routing:

- If 200, render the lectures for the section
- Else send to home page unauthorized

PUT /courses/:course_id/sections/:section_id/lectures/:lecture_id

User Action: teacher wants to publish or unpublish a lecture
Backend Action: authenticate token, update the lecture to either published or unpublished. If unpublished, unpublish all of the questions for the lecture and trigger enrollment rollups and question rollups for any that were still published
Backend Response:

- If auth and successfully updated, return 200
- If not found, return 404
- If internal error return 500
- If student 403

- Else 401

Routing:

- If 200 simply update the toggle appropriately and stay on the lectures page
- If 404, re-request lectures
- If 500 display error to try to publish again
- Else send to home page unauthenticated

…/questions/:question_id

PUT /courses/:course_id/lectures/:lecture_id/questions/:question_id

User Action: teacher want to (un)publish a question in the lecture
Backend Action: authenticate token, update the question in lecture. If the question is being unpublished, trigger a question rollup on the queue
Backend Response:

- If auth and successfully updated, return 200
- If not found return 404
- If internal error return 500
- If student 403
- Else 401

Routing:

- If 200, just stay on the page for the lecture
- If 404, re-request the lecture
- If 500, display error and say try again
- Else send to home page unauthorized

GET /courses/:course_id/lectures/:lecture_id/questions/:question_id

User Action: teacher wants to view a question inside a lecture
Backend Action: authenticate, find the question, and get the rollup information
Backend Response:

- If authenticated and everything can be found, return the info with 200
- If something can’t be found, return 404
- If student return 403
- Else return 401

Routing:

- If 200, render the question and scoring information
- If 404, re-request the questions and go to questions page
- Else send to home page unauthorized

…/responses

GET /courses/:course_id/sections/:section_id/lectures/:lecture_id/responses

User Action: a teacher wants to view all the student’s scores (accuracy and participation) for the lecture

GET /courses/:course_id/sections/:section_id/lectures/:lecture_id/responses/:enrollment_id

User Action: a teacher wants to view a student's responses to the questions in the lecture

/enrollments (teacher only)

GET /courses/:course_id/enrollments

User Action: teacher is requesting roster for a given course
Backend Action: authenticate token & fetch all enrollments, teachers and students excluding self
Backend Response:

- If authenticated, return the enrollments with 200
- Else, return 403

Routing:

- If 200, stay put and render all the enrollments in a list
- Else, send to home page unauthorized

DELETE /courses/:course_id/enrollments/:enrollment_id

User Action: teacher is manually removing a student or teacher from a given course
Backend Action: authenticate token & delete the enrollment & associated data
Backend Response:

- If authenticated, return a 200
- Else, return 403

Routing:

- If 200, stay on page & remove the element that was deleted from the list
- Else, send to home page unauthorized

PUT /courses/:course_id/enrollments/:enrollment_id

User Action: teacher is changing a student's section
Backend Action: authenticate the token & update the section if authenticated
Backend response:

- If authenticated, send enrollment data with 200 status
- Else return 403

Routing:

- If 200, update that enrollment element with the correct section in the display
- Else, send to home page unauthorized

/lectures

GET /courses/:course_id/lectures

User Action: user clicked on the lectures for the course they're in
Backend Action: authenticate the token and find the lectures (For teachers, all of them. For students only the published lectures - should be all past ones and any live ones. Live ones should have an indication in the response for students to know to enter)
Backend Response:

- If authenticated, return all of the lectures with 200 status
- Else return 403

Routing:
- If 200, render all the lectures on the lectures page
    - For students, any live lectures should be indicated as such
- Else, send to home page unauthorized

POST /courses/:course_id/lectures

Body: { “title”: “”, “description”: “”, “order”: “” }
User Action: teacher is creating a lecture for this course
Backend Action: authenticate the token, create the lecture, and associate the lecture to each section
Backend Response:
- If authenticated & lecture creation valid, return 200 with lecture information
- If lecture creation isn’t valid, return 404 with errors from validation
- Else return 403

Routing:
- If 200, go to the lecture page
- Else if 404, stay on page and display error
- Else send to home page unauthorized

/:lecture_id

PUT /courses/:course_id/lectures/:lecture_id

Body: { “title”: “”, “order”: “”, “description”: “” }
User Action: teacher submits lecture form
Backend Action: authenticate the token, then update the lecture
Backend Response:
- If authenticated & successful update, send 200 with the new lecture information
- If authenticated but validation issue, send 400 with errors
- Else return 403

Routing:
- If 200, go to the lecture page
- If 400, display errors on page
- Else return 403

GET /courses/:course_id/lectures/:lecture_id

User Action: user clicked on a lecture item from the lecture page
Backend Action: authenticate the token, then find the lecture and check if it’s published (only for students), then also get all the questions for that lecture (students only get published questions for published lectures)
Backend response:
- If authenticated as a teacher, return the lecture information with 200
- If authenticated as student & the lecture is published, return the lecture info with 200

- If authenticated as student & the lecture is not published, check if the lecture has been previously open and if so, return summary information with 200
- If authenticated as student & the lecture is not published & never has been, return 404
- Else return 403

Routing:
- If 200, render the proper lecture view depending on the role & publication status & questions publication status
- If 404, send the student or teacher back to the lectures page
- Else send to home page unauthorized

DELETE /courses/:course_id/lectures/:lecture_id

User Action: teacher is trying to delete a lecture
Backend Action: authenticate the token, then delete the lecture & all associations related
Backend Response:
- If authenticated and successful deletion, send 200
- If auth but deletion failure, send 500
- Else, return 403

Routing:
- If 200, send teachers to lectures page
- If 500, display error and try again
- Else send to home page unauthorized

/questions

PUT /courses/:course_id/lectures/:lecture_id/questions

Body: { “questions”: [“question_id”, “question_id”] }
User Action: teacher is updating the order of questions in a lecture or adding a question to a lecture
Backend Action: authenticate the token, then either change the order of at least 2 of the questions or just add the question that’s not already in the lecture. Or, perform a combination of the two
Backend Response:
- If authenticated and there weren’t any issues with updating the questions, return 200
- If authenticated but there was an error, return informative errors with 404 or perhaps 500 depending on what happened
- Else return 403

Routes:
- If 200, render the lecture page
- If 404, stay on the page and render the errors
- If 500, stay on the page and give notice to try again
- Else send to home page unauthorized

POST question/:question_id/responses

User Action: student is submitting his or her response to a question

Backend action: authenticate the token, then create a response
Backend Response:

- If authenticated and created, send 200
- If authenticated but the question is closed, send a 400
- If authenticated but something failed, send 500
- Else send 401

Routing:

- If 200, stay on question
- If 400, display question closed notice and send back to lecture page
- If 500, say something went wrong and to try again
- Else send to home unauthorized

PUT question/:question_id/responses/:response_id

User Action: student is resubmitting his or her response to a question
Backend action: authenticate the token, then find the response & update it
Backend Response:

- If authenticated and updated, send 200
- If authenticated but the question is closed, send a 400
- If authenticated but the question is not found, send a 404
- If authenticated but something failed, send 500
- Else send 401

Routing:

- If 200, stay on question
- If 400, display question closed notice and send back to lecture page
- If 404, display a not found, send to lecture page
- If 500, say something went wrong and to try again
- Else send to home unauthorized

/questions (teacher only)

GET /courses/:course_id/questions?search=””&page=””&perPage=””

Params

- Search: query string
- Page: page number
- perPage: questions per page

User Action: teacher wants to manage the question set for their course or browse questions to add to a lecture
Backend Action: authenticate the token, then fetch all questions for the course and search the stem field if search is passed in query parameter
Backend Response:

- If authenticated as a teacher, return the questions with 200
- Else return 403

Routing:

- If 200, stay on the page and render the questions
- Else send to home page unauthorized

POST /courses/:course_id/questions

Body: { “type”: “”, “stem”: “”, “content”: {}, “answers”: {} }
User Action: teacher has filled out new question and submits form
Backend Action: authenticate token, validate params and create a question
Backend Response:
- If authenticated as teacher & question creates successfully, return question content with 200
- If authenticated as teacher but question validation fails, return 400 with errors
- Else return 403

Routing:
- If 200, send user back to courses
- If 400, render validation errors and keep form
- Else send to home page unauthorized

/:question_id

PUT /courses/:course_id/questions/:question_id

Body: { “stem”: “”, “content”: {}, “answers”: {} }
User Action: teacher has edited question form
Backend Action: authenticate token & update question
Backend response:
- If authenticated as teacher & question updates successfully, return question content with 200
- If authenticated as teacher but question validation fails, return 400 with errors
- Else return 403

Routing:
- If 200, send user back to courses
- If 400, render validation errors and keep form
- Else send to home page unauthorized

Questions & Comments

Implementation

- Should the frontend by default send an authentication request every time it’s loaded?

Functionality

- How will teachers be invited to courses?
- Can a teacher who created the course remove other teachers from the course?

What Else?

- Work more thinking about any sort of summary data the teachers need